

Министерство науки и высшего образования Российской Федерации

ЧИТИНСКИЙ ИНСТИТУТ (ФИЛИАЛ)
ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО ОБРАЗОВАТЕЛЬНОГО
УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ
«БАЙКАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

УТВЕРЖДЕН

на заседании кафедры информационных
технологий и высшей математики
24 февраля 2025 г. протокол № 6

Заведующий кафедрой
Л.И. Трухина



**ОЦЕНОЧНЫЕ МАТЕРИАЛЫ
(ФОНД ОЦЕНОЧНЫХ СРЕДСТВ)
ПО УЧЕБНОЙ ДИСЦИПЛИНЕ**

Б1.О.21 Объектно-ориентированный анализ и программирование

Направление подготовки: 38.03.05 Бизнес-информатика

Направленность (профиль): Цифровая экономика

Квалификация выпускника: бакалавр

Чита, 2025 г.

**Структура
фонда оценочных средств
по дисциплине «Объектно-ориентированный анализ и программирование»**

№ п/п	Этапы формирования компетенций	Перечень формируемых компетенций	ЗУНы (З.1, У1, Н1...)	Контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и опыта деятельности, характеризующих этапы формирования компетенций в процессе освоения образовательной программы	Описание показателей и критериев оценивания компетенций на различных этапах формирования, описания шкал оценивания
1	Классы и объекты	ОПК-3	З. Знать, как разрабатывать объектно-ориентированные алгоритмы и программы, пригодные для практического применения; У. Уметь разрабатывать объектно-ориентированные алгоритмы и программы, пригодные для практического применения; Н. Владеть навыками разработки объектно-ориентированные алгоритмов и программ, пригодных для практического	Т, РЗ	5 баллов - тестирование; 10 баллов - решение задач для проверки умений; 10 баллов - решение задач для проверки навыков.

			применения.		
2	Наследование	ОПК-3	З. Знать, как разрабатывать объектно-ориентированные алгоритмы и программы, пригодные для практического применения; У. Уметь разрабатывать объектно-ориентированные алгоритмы и программы, пригодные для практического применения; Н. Владеть навыками разработки объектно-ориентированных алгоритмов и программ, пригодных для практического применения.	Т, РЗ	5 баллов - тестирование; 10 баллов - решение задач для проверки умений; 10 баллов - решение задач для проверки навыков.
3	Дополнительные инструменты объектно-ориентированного программирования	ОПК-3	З. Знать, как разрабатывать объектно-ориентированные алгоритмы и программы, пригодные для практического применения; У. Уметь разрабатывать объектно-ориентированные алгоритмы и программы, пригодные для практического	Т, РЗ	5 баллов - тестирование; 10 баллов - решение задач для проверки умений; 10 баллов - решение задач для проверки навыков.

			применения; Н. Владеть навыками разработки объектно-ориентированные алгоритмов и программ, пригодных для практического применения.		
4	Создание игры в объектно-ориентированном стиле	ОПК-3	З. Знать, как разрабатывать объектно-ориентированные алгоритмы и программы, пригодные для практического применения; У. Уметь разрабатывать объектно-ориентированные алгоритмы и программы, пригодные для практического применения; Н. Владеть навыками разработки объектно-ориентированные алгоритмов и программ, пригодных для практического применения.	Т, РЗ	5 баллов - тестирование; 10 баллов - решение задач для проверки умений; 10 баллов - решение задач для проверки навыков.
5	Итого по текущей аттестации	ОПК-3			Итого 100 баллов
6	Промежуточная аттестация	ОПК-3	З. Знать, как разрабатывать объектно-ориентированные	Т, РЗ	20 баллов - тестирование; 40 баллов - решение

			алгоритмы и программы, пригодные для практического применения; У. Уметь разрабатывать объектно-ориентированные алгоритмы и программы, пригодные для практического применения; Н. Владеть навыками разработки объектно-ориентированные алгоритмов и программ, пригодных для практического применения.		задач для проверки навыков; 40 баллов - решение задач для проверки навыков.
				Итого	100

Министерство науки и высшего образования Российской Федерации
ЧИТИНСКИЙ ИНСТИТУТ (ФИЛИАЛ)
ФГБОУ ВО «БАЙКАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Кафедра информационных технологий и высшей математики

Оценочные средства текущего контроля:

Тема 1.1. Тест, решение задач (Приложение 1)

Тема 1.2. Тест, решение задач (Приложение 2)

Тема 1.3. Тест, решение задач (Приложение 3)

Тема 1.4. Тест, решение задач (Приложение 4)

Оценочные средства промежуточного контроля:

Материалы для промежуточного контроля в виде Экзамена в семестре 2.2.
(Приложение 5)

Билеты к экзамену во 2-м семестре на 2-м курсе
(материалы к экзамену приведены в Приложении 5)

Министерство науки и высшего образования
Российской Федерации
Читинский институт (филиал)
Федерального государственного бюджетного
образовательного учреждения
высшего образования
«БАЙКАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ»
(ЧИ ФГБОУ ВО «БГУ»)

Направление - 38.03.05
Бизнес-информатика
Профиль - Цифровая экономика
Кафедра информационных
технологий и высшей математики
Дисциплина –
Объектно-ориентированный анализ
и программирование

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ

1. Тест.
2. Две случайно выбранные задачи для проверки умений.
3. Две случайно выбранные задачи для проверки навыков.

Составитель _____ С.В. Бочкарев
Заведующий кафедрой _____ Л.И. Трухина

Приложение 1. Материалы для текущего контроля по теме 1.1.

1. Вопросы для проверки знаний:

1. Что такое класс в Python?

- a) Функция, которая возвращает объект
- b) Шаблон для создания объектов
- c) Модуль, содержащий переменные
- d) Библиотека для работы с данными

2. Как создать объект класса Person?

- a) Person.new()
- b) Person()
- c) new Person()
- d) create Person

3. Что такое атрибут объекта?

- a) Функция внутри класса
- b) Переменная, принадлежащая объекту
- c) Глобальная переменная
- d) Аргумент метода

4. Как объявить метод класса?

- a) Используя ключевое слово method
- b) Определив функцию внутри класса
- c) Через декоратор @method
- d) Только через лямбда-функцию

5. Что выведет код?

```
class Dog:
    sound = "Гав"
d = Dog()
print(d.sound)
```

- a) Гав
- b) Ошибка
- c) None
- d) ""

6. Как сделать атрибут приватным?

- a) Добавить @private перед ним
- b) Начать имя с двойного подчеркивания __
- c) Использовать ключевое слово private
- d) Никак, в Python нет приватных атрибутов

7. Как обратиться к приватному атрибуту `__name` извне?

- a) `obj.name`
- b) `obj.private.name`
- c) `obj._Person__name` (name mangling)
- d) Никак

8. Что такое `self` в методах класса?

- a) Это ссылка на класс
- b) Это ссылка на текущий объект
- c) Это обязательный аргумент для статических методов
- d) Это зарезервированное слово, но его можно заменить

9. Как создать атрибут класса (не объекта)?

- a) Внутри метода с `@classmethod`
- b) Определить его внутри класса, но вне методов
- c) Через `global`
- d) Только через `setattr()`

10. Что такое инкапсуляция?

- a) Наследование классов
- b) Скрытие внутренней реализации и защита данных
- c) Перегрузка операторов
- d) Динамическое изменение типов

11. Что выведет код?

```
class Car:
    color = "Красный"
c1 = Car()
c2 = Car()
c2.color = "Синий"
print(c1.color)
```

- a) Синий
- b) Красный
- c) Ошибка
- d) None

12. Какой метод вызывается при создании объекта?

- a) `__new__`
- b) `__init__`
- c) `__create__`
- d) `__start__`

13. Как запретить добавление новых атрибутов объекту?

- a) `@no_new_attrs`
- b) `__slots__`
- c) `@final`
- d) `@sealed`

14. Как проверить, является ли объект экземпляром класса?

- a) `obj.isinstance(Class)`
- b) `isinstance(obj, Class)`
- c) `obj == Class`
- d) `type(obj) is Class`

15. Что делает `@property`?

- a) Делает метод приватным
- b) Позволяет использовать метод как атрибут
- c) Запрещает изменение атрибута
- d) Автоматически вызывает метод

16. Какой метод вызывается при удалении объекта?

- a) `__del__`
- b) `__destroy__`
- c) `__del__` (но не гарантируется вызов)
- d) `__exit__`

17. Как сделать метод класса (не объекта)?

- a) `@objectmethod`
- b) `@classmethod`
- c) `@sharedmethod`
- d) `@staticmethod`

18. Что выведет код?

```
class A:  
    def __init__(self, x):  
        self.x = x
```

```
a = A(5)
```

```
print(a.y)
```

- a) 5
- b) None
- c) Ошибка
- d) 0

19. Как добавить атрибут объекту после создания?

- a) `obj.attr = value`
- b) Только через `__dict__`

- c) Только в `__init__`
- d) Никак

20. Какой принцип ООП нарушает прямой доступ к атрибутам?

- a) Наследование
- b) Инкапсуляция
- c) Полиморфизм
- d) Абстракция

21. Какой метод вызывается при `print(obj)`?

- a) `__str__`
- b) `__print__`
- c) `__repr__`
- d) `__display__`

22. Какой атрибут содержит все методы и поля класса?

- a) `__methods__`
- b) `__dict__`
- c) `__all__`
- d) `__slots__`

23. Какой метод позволяет использовать объект как функцию?

- a) `__func__`
- b) `__call__`
- c) `__run__`
- d) `__execute__`

24. Как динамически добавить метод классу?

- a) `class.add_method()`
- b) Присвоить функцию атрибуту класса
- c) Через `@dynamic`
- d) Никак

25. Как проверить, есть ли атрибут у объекта?

- a) `obj.has_attr()`
- b) `hasattr(obj, 'attr')`
- c) `'attr' in obj`
- d) `obj.attr.exists()`

2. Задачи для проверки умений:

Задача 1. Класс Person

Создайте класс Person с атрибутами name и age. Добавьте метод introduce(), который выводит: "Меня зовут [name], мне [age] лет."

Задача 2. Класс Rectangle

Создайте класс Rectangle с атрибутами width и height. Добавьте методы:

area() — возвращает площадь.

perimeter() — возвращает периметр.

Задача 3. Приватный атрибут в BankAccount

Модифицируйте BankAccount, сделав balance приватным (__balance).

Добавьте методы get_balance() и set_balance(amount) (с проверкой: amount ≥ 0).

Задача 4. Класс Temperature с конвертацией

Создайте класс Temperature с приватным атрибутом _celsius. Добавьте:

Свойство celsius (геттер/сеттер).

Свойство fahrenheit (геттер/сеттер, конвертация).

Задача 5. Класс Library и Book

Создайте класс Library, который хранит список книг (Book).

Задача 6. Класс Shop и Product

Создайте класс Shop, который может добавлять и продавать товары (Product).

Задача 7. Класс Hospital и Patient

Создайте класс Hospital, который регистрирует пациентов (Patient).

3. Задачи для проверки навыков:

Задача 1. Класс BankAccount

Создайте класс BankAccount с атрибутом balance (начальный баланс = 0).

Реализуйте методы:

deposit(amount) — пополнение счёта.

withdraw(amount) — снятие денег (если средств достаточно).

Задача 2. Класс Student

Создайте класс Student с атрибутами name и grades (список оценок). Добавьте метод average_grade(), возвращающий средний балл.

Задача 3. Класс Car

Создайте класс Car с атрибутами brand, model, year. Добавьте метод get_info(), возвращающий строку с данными.

Задача 4. Класс Password с валидацией

Создайте класс Password с приватным атрибутом __password. Реализуйте:

Метод set_password(pwd) (только если длина ≥ 8 символов).

Метод check_password(pwd) (проверяет соответствие).

Задача 5. Класс Person с валидацией возраста

Создайте класс `Person` с приватным атрибутом `__age` (только ≥ 0). Добавьте свойство `age` с геттером/сеттером.

Задача 6. Класс `Student` с защищёнными оценками

Создайте класс `Student` с приватным списком `__grades`. Добавьте:

Метод `add_grade(grade)` (добавляет оценку 1-10).

Метод `get_grades()` (возвращает копию списка).

Задача 7. Класс `University` и `Student`

Создайте класс `University`, который управляет студентами (`Student`).

Задача 8. Класс `Playlist` и `Song`

Создайте класс `Playlist`, который хранит песни (`Song`).

Приложение 2. Материалы для текущего контроля по теме 1.2.

1. Вопросы для проверки знаний:

1. Что такое наследование в Python?

- a) Копирование кода из одного класса в другой
- b) Механизм, позволяющий классу наследовать атрибуты и методы другого класса
- c) Способ объединения нескольких классов в один модуль
- d) Процесс создания экземпляра класса

2. Какой метод вызывается при создании экземпляра дочернего класса?

- a) `__init__` дочернего класса, а если его нет – `__init__` родительского
- b) Только `__init__` родительского класса
- c) `__new__` и `__init__` дочернего класса
- d) Никакой метод не вызывается автоматически

3. Как вызвать метод родительского класса из дочернего?

- a) `Parent.method(self)`
- b) `super().method()`
- c) `self.parent.method()`
- d) `super(Parent, self).method()`

4. Что такое множественное наследование?

- a) Наследование одного класса от нескольких модулей
- b) Наследование одного класса от нескольких родительских классов
- c) Создание нескольких экземпляров одного класса
- d) Наследование от абстрактного класса

5. Как Python обрабатывает порядок наследования при множественном наследовании?

- a) Случайным образом
- b) По алфавиту
- c) С помощью MRO (Method Resolution Order)
- d) По порядку объявления родителей

6. Как можно посмотреть порядок разрешения методов (MRO) класса?

- a) `help(class)`
- b) `class.__mro__` или `class.mro()`
- c) `dir(class)`
- d) `inspect.getmro(class)`

7. Что будет, если два родительских класса имеют метод с одинаковым именем?

- a) Произойдёт ошибка
- b) Будет вызван метод первого родителя в MRO
- c) Будет вызван метод последнего родителя в MRO
- d) Оба метода будут вызваны

8. Что такое переопределение метода?

- a) Удаление метода родительского класса
- b) Создание метода в дочернем классе с тем же именем, что и у родителя
- c) Изменение сигнатуры метода родителя
- d) Копирование метода из другого класса

9. Может ли класс наследоваться от встроенных типов (например, list, dict)?

- a) Да
- b) Нет
- c) Только если переопределить `__new__`
- d) Только в Python 2

10. Как проверить, является ли класс подклассом другого?

- a) `isinstance(obj, Parent)`
- b) `issubclass(Child, Parent)`
- c) `Child in Parent.subclasses()`
- d) `hasattr(Child, Parent)`

11. Можно ли динамически изменить родительский класс объекта?

- a) Да, через `obj.__class__ = NewClass`
- b) Нет, наследование определяется при создании класса
- c) Только через `__bases__`
- d) Только в Python 2

12. Что делает `super().__init__()` в дочернем классе?

- a) Ничего, это синтаксическая ошибка
- b) Вызывает `__init__` родительского класса
- c) Создаёт новый экземпляр родителя
- d) Удаляет экземпляр родителя

13. Какой из вариантов демонстрирует корректное множественное наследование?

- a) `class Child(Parent1, Parent2):`
- b) `class Child(Parent1)(Parent2):`
- c) `class Child inherits Parent1, Parent2:`

d) `class Child: Parent1, Parent2`

14. Что произойдёт, если в цепочке наследования метод не найден?

- a) Будет вызван метод `object.__missing__`
- b) Возникнет `AttributeError`
- c) Python автоматически создаст пустой метод
- d) Будет вызван метод `__getattr__`

15. Можно ли унаследоваться от класса, определённого в другом модуле?

- a) Да
- b) Нет, только из того же модуля
- c) Только если импортировать его через `import *`
- d) Только если родительский класс — публичный

16. Как запретить добавление новых атрибутов в дочернем классе?

- a) Использовать `@final`
- b) Определить `__slots__` в родительском классе
- c) Сделать все атрибуты приватными
- d) Переопределить `__setattr__`

17. Что выведет `print(Child.__mro__)`, если `Child(Parent1, Parent2)`?

- a) Список всех методов класса
- b) Кортеж с порядком разрешения методов (от `Child` до `object`)
- c) Имена всех родительских классов в случайном порядке
- d) `None`

18. Как проверить, был ли метод переопределён в дочернем классе?

- a) `Child.method == Parent.method`
- b) `Child.method is not Parent.method`
- c) `hasattr(Child, 'overridden')`
- d) `inspect.ismethod(Child.method)`

19. Какой метод позволяет запретить удаление атрибута в дочернем классе?

- a) `__lockattr__`
- b) `__delattr__`
- c) `__slots__`
- d) `__delete__`

20. Можно ли динамически добавить родительский класс?

- a) Да, через `Child.__bases__ += (NewParent,)`
- b) Нет, после создания класса это невозможно
- c) Только через `metaclass`
- d) Только в Python 2

21. Что будет, если в `__init__` дочернего класса не вызвать `super().__init__()`?

- a) Экземпляр не создастся
- b) `__init__` родителя не выполнится, если он не вызван явно
- c) Произойдёт рекурсия
- d) Python вызовет его автоматически

22. Как создать класс, экземпляры которого нельзя изменить?

- a) Сделать все атрибуты приватными
- b) Использовать `__slots__` и не определять `__setattr__`
- c) Наследоваться от `tuple`
- d) Запретить наследование

23. Какой метод вызовется при `obj.method()`, если `method` есть в родителе, но не переопределён?

- a) Метод родительского класса
- b) `AttributeError`
- c) `object.method`
- d) Ничего, синтаксическая ошибка

24. Что будет, если у двух родителей есть атрибут с одним именем?

- a) Ошибка при создании класса
- b) Будет использован атрибут первого родителя в MRO
- c) Будет создан новый атрибут
- d) `AttributeError` при обращении

25. Как получить список всех родительских классов?

- a) `dir(Child)`
- b) `Child.__bases__`
- c) `Child.parents()`
- d) `help(Child)`

2. Задачи для проверки умений:

Задача 1. Переопределение `__init__`

Создайте класс `Person` с `__init__` (имя, возраст) и класс `Employee`, добавляющий `salary`.

Задача 2. Наследование от встроенных классов

Создайте класс `MyList`, наследующий `list`, добавляющий метод `print_all()`.

Задача 3. Проблема ромбовидного наследования

Создайте иерархию `A -> B`; `A -> C`; `B, C -> D`. Убедитесь, что `A.__init__` вызывается 1 раз.

Задача 4. Наследование с добавлением нового метода

Создайте класс `Animal` с методом `eat()` (выводит "Eating..."). Затем создайте класс `Bird`, который наследует `Animal` и добавляет метод `fly()` (выводит "Flying").

Задача 5. Переопределение метода без `super()`

Создайте класс `Vehicle` с методом `move()` ("Moving..."). Затем создайте класс `Car`, который полностью переопределяет `move()` ("Driving a car").

Задача 6. Множественное наследование с общим методом

Создайте классы `A` (метод `show()`: "A") и `B` (метод `show()`: "B"). Затем создайте класс `C(A, B)` и вызовите `show()`. Объясните результат.

Задача 7. Множественное наследование с разными методами

Создайте классы `X` (метод `a()`: "X.a") и `Y` (метод `b()`: "Y.b"). Затем создайте класс `Z(X, Y)` и вызовите оба метода.

3. Задачи для проверки навыков:

Задача 1. Наследование и добавление атрибута

Создайте класс `Book` с `__init__` (принимает `title`). Затем создайте класс `EBook`, который добавляет атрибут `format`.

Задача 2. Наследование и вызов метода из родителя

Создайте класс `Alpha` с методом `run()` ("Alpha runs"). Затем создайте класс `Beta`, который вызывает `run()` из `Alpha`.

Задача 3. Наследование и изменение списка

Создайте класс `Parent` с атрибутом `items = [1, 2]`. Затем создайте класс `Child`, который добавляет 3 в `items`.

Задача 4. Иерархия классов с многоуровневым наследованием

Создайте цепочку наследования:

`A` (метод `test()`: "A")

`B(A)` (переопределяет `test()`: "B")

`C(B)` (переопределяет `test()`: "C")

`D(C)` (вызывает `test()` всех родителей через `super()`).

Задача 5. Множественное наследование с пересекающимися методами

Создайте классы:

`X` (метод `process()`: "X.process")

`Y` (метод `process()`: "Y.process")

`Z(X, Y)` (вызывает `process()` через `super()`).

Задача 6. Множественное наследование с инициализацией

Создайте классы:

Alpha (`__init__` принимает x, сохраняет в `self.x`)

Beta (`__init__` принимает y, сохраняет в `self.y`)

Gamma(Alpha, Beta) (инициализирует оба родителя).

Задача 7. Наследование и вызов методов из разных родителей

Создайте классы:

A (метод `start()`: "A.start")

B (метод `stop()`: "B.stop")

C(A, B) (метод `restart()` вызывает `stop()`, затем `start()`).

Задача 8. Множественное наследование с общим предком

Создайте иерархию:

Animal (метод `sound()`: "Какой-то звук")

Dog(Animal) (переопределяет `sound()`: "Гав")

Cat(Animal) (переопределяет `sound()`: "Мяу")

Pet(Dog, Cat) (вызывает `sound()` через `super()`).

Приложение 3. Материалы для текущего контроля по теме 1.3.

1. Вопросы для проверки знаний:

1. Что такое полиморфизм в ООП?

- a) Возможность изменять код во время выполнения
- b) Способность объектов с одинаковым интерфейсом иметь разную реализацию
- c) Объединение данных и методов в одном классе
- d) Наследование свойств от родительского класса

2. Какой принцип ООП позволяет вызывать один и тот же метод для разных объектов?

- a) Инкапсуляция
- b) Полиморфизм
- c) Наследование
- d) Абстракция

3. Как полиморфизм реализуется в Python?

- a) Только через наследование
- b) Через наследование и перегрузку операторов
- c) Только через абстрактные классы
- d) Через статические методы

4. Можно ли считать полиморфизмом перегрузку операторов?

- a) Нет, это разные концепции
- b) Да, потому что один оператор может работать по-разному для разных классов
- c) Только если используется наследование
- d) Только для числовых типов

5. Как обозначается статический метод в Python?

- a) `@classmethod`
- b) `@staticmethod`
- c) `@property`
- d) `@abstractmethod`

6. Чем отличается статический метод от обычного?

- a) Он не принимает `self` и принадлежит классу, а не экземпляру
- b) Он может изменять состояние класса
- c) Он доступен только из других методов класса
- d) Он автоматически вызывается при создании объекта

7. Можно ли обращаться к статическому методу через экземпляр класса?

- a) Нет, только через класс
- b) Да, но это не рекомендуется
- c) Только если метод объявлен как `@classmethod`
- d) Только если класс наследуется от ABC

8. Какой метод перегружает оператор +?

- a) `__add__`
- b) `__plus__`
- c) `__sum__`
- d) `__add__` или `__radd__`

9. Как перегрузить оператор сравнения ==?

- a) `__cmp__`
- b) `__eq__`
- c) `__equals__`
- d) `__compare__`

10. Какой метод перегружает оператор индексирования []?

- a) `__getitem__`
- b) `__setitem__`
- c) `__index__`
- d) `__getitem__` и `__setitem__`

11. Какой декоратор помечает метод как абстрактный?

- a) `@abstract`
- b) `@abstractmethod`
- c) `@virtualmethod`
- d) `@mustoverride`

12. Можно ли создать экземпляр абстрактного класса?

- a) Да, если в нём нет абстрактных методов
- b) Нет, Python выдаст ошибку
- c) Только если класс наследуется от `object`
- d) Да, но только через `@staticmethod`

13. Можно ли сочетать `@staticmethod` и `@abstractmethod`?

- a) Да, без ограничений
- b) Нет, абстрактный метод должен иметь `self`
- c) Только если метод возвращает `None`
- d) Только в классах без наследования

14. Какой из этих методов не является магическим?

- a) `__len__`
- b) `__iter__`
- c) `__static__`
- d) `__contains__`

15. Какой метод перегружает оператор умножения *?

- a) `__mult__`
- b) `__mul__`
- c) `__times__`
- d) `__product__`

16. Как перегрузить оператор +=?

- a) `__plus_equals__`
- b) `__iadd__`
- c) `__aplus__`
- d) `__add_assign__`

17. Как перегрузить оператор in?

- a) `__has__`
- b) `__contains__`
- c) `__in__`
- d) `__include__`

18. Можно ли добавить неабстрактные методы в абстрактный класс?

- a) Нет, только абстрактные
- b) Да, они будут доступны в дочерних классах
- c) Только если класс наследуется от `object`
- d) Только статические методы

19. Какой декоратор делает весь класс абстрактным?

- a) `@abstractclass`
- b) Класс должен наследоваться от `ABC`
- c) `@abstractmethod`
- d) `@force_abstract`

20. Что произойдёт, если в дочернем классе не переопределить все абстрактные методы?

- a) Ошибка при создании экземпляра
- b) Методы останутся пустыми
- c) Класс автоматически станет абстрактным
- d) Ничего, Python проигнорирует это

21. Можно ли обратиться к статическому методу из другого статического метода?

- a) Нет, статические методы изолированы
- b) Да, через имя класса
- c) Только если методы объявлены в одном модуле
- d) Только через `@classmethod`

22. Зачем использовать `@staticmethod` вместо обычной функции вне класса?

- a) Для логической группировки кода
- b) Чтобы метод был приватным
- c) Для ускорения работы программы
- d) Чтобы запретить наследование

23. Какой принцип ООП не связан напрямую с полиморфизмом?

- a) Инкапсуляция
- b) Наследование
- c) Абстракция
- d) Все связаны

24. Как создать абстрактный класс?

- a) `@abstract`
- b) `from abc import ABC, abstractmethod` + наследование от ABC
- c) `class Abstract`
- d) `@abstractclass`

25. Что такое "интерфейс" в Python?

- a) Специальный синтаксис для наследования
- b) Абстрактный класс, где все методы — `@abstractmethod`
- c) Декоратор `@interface`
- d) Наследование от `Interface`

2. Задачи для проверки умений:

Задача 1. Животные и их звуки

Создайте классы `Dog` и `Cat` с методом `make_sound()`. Используйте полиморфизм, чтобы вызвать этот метод для объектов разных классов.

Задача 2. Фигуры с методом `draw()`

Создайте классы `Circle` и `Square` с методом `draw()`, который возвращает строку-описание фигуры. Продемонстрируйте полиморфизм.

Задача 3. Калькулятор

Создайте класс `Calculator` со статическими методами для сложения, вычитания и умножения.

Задача 4. Вектор с перегрузкой +

Создайте класс Vector с перегруженным оператором + для сложения векторов.

Задача 5. Фигуры с методом area()

Создайте абстрактный класс Shape с методом area(). Реализуйте его в Circle и Rectangle.

Задача 6. Абстрактный транспорт

Создайте абстрактный класс Transport с методом move(). Реализуйте его в Car и Boat.

Задача 7. Банковский счёт с перегрузкой операторов

Создайте класс BankAccount с перегрузкой + (пополнение) и - (снятие), а также статическим методом для проверки суммы.

3. Задачи для проверки навыков:

Задача 1. Игровые персонажи

Создайте классы Warrior и Mage с методом attack(), возвращающим разный урон (10 и 15 соответственно).

Задача 2. Валидация email

Добавьте в класс User статический метод validate_email(), который проверяет, содержит ли email символ @.

Задача 3. Кеширование данных

Создайте класс Cache со статическим словарём для хранения данных и методами set(key, value) и get(key).

Задача 4. Перегрузка == для класса Book

Перегрузите оператор == для класса Book, чтобы книги сравнивались по названию и автору.

Задача 5. Перегрузка * для матриц

Создайте класс Matrix с перегруженным оператором * для умножения матриц.

Задача 6. Абстрактный класс Animal

Создайте абстрактный класс Animal с методами eat() и sleep(). Реализуйте его в Lion и Fish.

Задача 7. Генератор случайных чисел с полиморфизмом

Создайте классы RandomInt и RandomFloat с методом generate(), возвращающим случайное число. Используйте полиморфизм.

Задача 8. Абстрактный класс Device с перегрузкой ==

Создайте абстрактный класс Device с методом turn_on() и перегрузите == для сравнения по мощности. Реализуйте в Lamp и TV.

Приложение 4. Материалы для текущего контроля по теме 1.4.

1. Вопросы для проверки знаний:

1. Какой метод в Pygame вызывается для обновления экрана?

- A) `pygame.redraw()`
- B) `pygame.update_screen()`
- C) `pygame.display.update()`
- D) `pygame.screen.refresh()`

2. Как загрузить изображение для спрайта в Pygame?

- A) `pygame.open_image("sprite.png")`
- B) `pygame.image.load("sprite.png")`
- C) `pygame.load("sprite.png")`
- D) `pygame.sprite("sprite.png")`

3. Как создать анимацию персонажа, используя несколько изображений?

- A) Загрузить одно изображение и масштабировать его
- B) Загрузить список изображений и переключать их с задержкой
- C) Использовать только векторную графику
- D) Анимация невозможна в Pygame

4. Какой метод отвечает за обработку столкновений объектов в Pygame?

- A) `check_overlap()`
- B) `detect_collision()`
- C) `collidect()`
- D) `intersects()`

5. Как сохранить текущий счёт игрока в файл?

- A) `save(score, "score.txt")`
- B) `with open("score.txt", "w") as f: f.write(str(score))`
- C) `file.write(score, "score.txt")`
- D) `export score to "score.txt"`

6. Какой метод в Pygame используется для обработки событий клавиатуры?

- A) `pygame.key_press()`
- B) `pygame.event.get()`
- C) `pygame.input.keyboard()`
- D) `pygame.keyboard.event()`

7. Какой класс в Pygame отвечает за работу с изображениями и спрайтами?

- A) `pygame.Graphics`
- B) `pygame.Draw`
- C) `pygame.Surface`
- D) `pygame.Texture`

8. Какой метод вызывается для отрисовки спрайта на экране?

- A) `pygame.render()`
- B) `screen.blit(sprite, (x, y))`
- C) `pygame.draw_sprite()`
- D) `sprite.display()`

9. Как создать несколько экземпляров врагов в игре?

- A) `enemy = Enemy()`
- B) `enemies = [Enemy() for _ in range(10)]`
- C) Враги не могут создаваться программно
- D) Только вручную через `pygame.add_enemy()`

10. Какой метод используется для проверки столкновения пули и врага?

- A) `bullet.touch(enemy)`
- B) `pygame.sprite.collide_rect(bullet, enemy)`
- C) `if bullet == enemy:`
- D) `bullet.hit_test(enemy)`

11. Как загрузить фоновую музыку в Pygame?

- A) `pygame.load_song("music.mp3")`
- B) `pygame.mixer.music.load("music.mp3")`
- C) `pygame.audio.load("music.mp3")`
- D) `pygame.soundtrack("music.mp3")`

12. Как отображать счет игрока на экране?

- A) `print(score)`
- B) `font.render(f"Score: {score}", True, (255, 255, 255))`
- C) `pygame.show_text(score)`
- D) Только через консоль

13. Как создать меню с кнопками "Start" и "Quit"?

- A) Только через консоль
- B) Отображать прямоугольники (`Rect`) и проверять клики мыши
- C) `pygame.autocreate_menu()`
- D) Меню нельзя сделать в Pygame

14. Как реализовать систему уровней в игре?

- A) Создать один огромный уровень
- B) Загружать разные карты или параметры при переходе на новый уровень
- C) Уровни нельзя запрограммировать
- D) Только через внешние программы

15. Как реализовать "инвентарь" игрока в ООП-стиле?

- A) Создать глобальный список
- B) Сделать класс `Inventory` с методами `add_item()`, `remove_item()`
- C) Хранить предметы в переменных `item1`, `item2`...
- D) Инвентарь невозможен в Pygame

16. Как сделать, чтобы игра сохраняла прогресс при закрытии?

- A) Прогресс нельзя сохранить
- B) Записать данные (уровень, HP, инвентарь) в JSON или файл
- C) Использовать `pygame.auto_save()`
- D) Хранить всё в оперативной памяти

17. Как реализовать "подбор предметов" (коллизия игрока и предмета)?

- A) Удалять все предметы на карте
- B) Проверять `colliderect()` и вызывать метод `player.pick_item(item)`
- C) Предметы нельзя подбирать
- D) Использовать `pygame.collect()`

18. Какой тип данных лучше всего подходит для хранения анимаций персонажа?

- A) Строка (`str`)
- B) Список (`list`) из Surface-объектов
- C) Число (`int`)
- D) Словарь (`dict`) без изображений

19. Как сделать "искусственную задержку" для атаки бота?

- A) Бот должен атаковать каждый кадр
- B) Использовать `pygame.time.get_ticks()` для контроля времени
- C) `time.sleep(1)`
- D) Только через многопоточность

20. Как обработать столкновение пули со стеной?

- A) Пуля проходит сквозь стены
- B) При `colliderect()` удалять пулю из списка
- C) Стены нельзя проверить на коллизии
- D) Использовать `pygame.bullet_stop()`

21. Как реализовать "разные типы урона" (огонь, лед и т.д.)?

- A) Сделать одну переменную `damage`
- B) Создать класс `Damage` с полями `type` и `value`
- C) Типы урона невозможны
- D) Использовать `pygame.damage_type()`

22. Как сделать, чтобы бот "патрулировал" зону?

- A) Бот стоит на месте
- B) Задать точки патрулирования и двигаться между ними
- C) Использовать `pygame.patrol_mode()`
- D) Только через ИИ с ИИ-библиотеками

23. Как реализовать "диалоговые окна" в игре?

- A) Выводить текст в консоль
- B) Создать класс `DialogBox` с отрисовкой текста и кнопок
- C) `pygame.show_dialog()`
- D) Диалоги невозможны

24. Какой метод ООП позволит игроку и боту наследовать общую логику движения?

- A) Композиция
- B) Создать родительский класс Character с методом move()
- C) Копировать код для каждого объекта
- D) Только через миксины

25. Как оптимизировать игру, если объектов слишком много?

- A) Не оптимизировать
- B) Использовать пространственное разделение (например, квадродерево)
- C) Удалить половину объектов
- D) `pygame.optimize_all()`

2. Задачи для проверки умений:

Задача 1. Создание класса игрока

Создайте класс Player, который имеет:

Атрибуты: x, y, speed, image (загруженное изображение).

Метод draw(), отрисовывающий игрока на экране.

Задача 2. Движение игрока по клавишам

Добавьте в класс Player метод update(), который перемещает объект по нажатию WASD.

Задача 3. Загрузка спрайта вместо квадрата

Замените `pygame.Surface` на загрузку изображения из файла `player.png`.

Задача 4. Создание класса врага

Создайте класс Enemy, который движется к игроку.

Задача 5. Столкновения игрока и врага

Добавьте проверку коллизии между Player и Enemy. При столкновении выводите "Game Over".

Задача 6. Анимация бега игрока

Загрузите 4 кадра анимации и переключайте их при движении.

Задача 7. Создание пуль

При нажатии SPACE создавайте пулю, которая летит вперед.

3. Задачи для проверки навыков:

Задача 1. Удаление пуль за пределами экрана

Если пуля вышла за границы экрана, удалите её из списка.

Задача 2. Система здоровья (HP)

Добавьте игроку $hp = 100$. При столкновении с врагом отнимайте 10 HP.

Задача 3. Сохранение рекорда в файл

При завершении игры сохраняйте счет в `score.txt`.

Задача 4. Случайное движение врагов

Реализуйте движение врагов случайным образом с изменением направления каждые 2 секунды.

Задача 5. Меню паузы

При нажатии P ставьте игру на паузу с выводом текста "Paused".

Задача 6. Подбор предметов (монет)

Создайте класс `Coin`. При столкновении с игроком увеличивайте счет.

Задача 7. Параллакс-фон

Реализуйте фон с 2 слоями, движущимися с разной скоростью.

Задача 8. Диалоговая система с NPC

Создайте класс `NPC`, который:

При приближении игрока выводит текст диалога.

Диалог отображается в прямоугольнике на экране.

Приложение 5.
Материалы для промежуточного контроля
в виде Экзамена в семестре 2.2.

ВОПРОСЫ ДЛЯ ПРОВЕРКИ ЗНАНИЙ

Компетенция ОПК-3: Способен управлять процессами создания и использования продуктов и услуг в сфере информационно-коммуникационных технологий, в том числе разрабатывать алгоритмы и программы для их практической реализации.

Знания: знать, как разрабатывать объектно-ориентированные алгоритмы и программы, пригодные для практического применения.

1. Что такое класс в Python?

- a) Функция, которая возвращает объект
- b) Шаблон для создания объектов
- c) Модуль, содержащий переменные
- d) Библиотека для работы с данными

2. Как создать объект класса Person?

- a) Person.new()
- b) Person()
- c) new Person()
- d) create Person

3. Что такое атрибут объекта?

- a) Функция внутри класса
- b) Переменная, принадлежащая объекту
- c) Глобальная переменная
- d) Аргумент метода

4. Как объявить метод класса?

- a) Используя ключевое слово method
- b) Определив функцию внутри класса
- c) Через декоратор @method
- d) Только через лямбда-функцию

5. Что выведет код?

```
class Dog:
    sound = "Гав"
d = Dog()
print(d.sound)
```

- a) Гав
- b) Ошибка
- c) None

d) ""

6. Как сделать атрибут приватным?

- a) Добавить @private перед ним
- b) Начать имя с двойного подчеркивания __
- c) Использовать ключевое слово private
- d) Никак, в Python нет приватных атрибутов

7. Как обратиться к приватному атрибуту __name извне?

- a) obj.name
- b) obj.private.name
- c) obj._Person__name (name mangling)
- d) Никак

8. Что такое self в методах класса?

- a) Это ссылка на класс
- b) Это ссылка на текущий объект
- c) Это обязательный аргумент для статических методов
- d) Это зарезервированное слово, но его можно заменить

9. Как создать атрибут класса (не объекта)?

- a) Внутри метода с @classattr
- b) Определить его внутри класса, но вне методов
- c) Через global
- d) Только через setattr()

10. Что такое инкапсуляция?

- a) Наследование классов
- b) Соккрытие внутренней реализации и защита данных
- c) Перегрузка операторов
- d) Динамическое изменение типов

11. Что выведет код?

```
class Car:
    color = "Красный"
c1 = Car()
c2 = Car()
c2.color = "Синий"
print(c1.color)
```

- a) Синий
- b) Красный
- c) Ошибка
- d) None

12. Какой метод вызывается при создании объекта?

- a) __new__
- b) __init__
- c) __create__

d) `__start__`

13. Как запретить добавление новых атрибутов объекту?

- a) `@no_new_attrs`
- b) `__slots__`
- c) `@final`
- d) `@sealed`

14. Как проверить, является ли объект экземпляром класса?

- a) `obj.isinstance(Class)`
- b) `isinstance(obj, Class)`
- c) `obj == Class`
- d) `type(obj) is Class`

15. Что делает `@property`?

- a) Делает метод приватным
- b) Позволяет использовать метод как атрибут
- c) Запрещает изменение атрибута
- d) Автоматически вызывает метод

16. Какой метод вызывается при удалении объекта?

- a) `__del__`
- b) `__destroy__`
- c) `__del__` (но не гарантируется вызов)
- d) `__exit__`

17. Как сделать метод класса (не объекта)?

- a) `@objectmethod`
- b) `@classmethod`
- c) `@sharedmethod`
- d) `@staticmethod`

18. Что выведет код?

```
class A:  
    def __init__(self, x):  
        self.x = x
```

```
a = A(5)  
print(a.y)
```

- a) 5
- b) None
- c) Ошибка
- d) 0

19. Как добавить атрибут объекту после создания?

- a) `obj.attr = value`
- b) Только через `__dict__`

- c) Только в `__init__`
- d) Никак

20. Какой принцип ООП нарушает прямой доступ к атрибутам?

- a) Наследование
- b) Инкапсуляция
- c) Полиморфизм
- d) Абстракция

21. Какой метод вызывается при `print(obj)`?

- a) `__str__`
- b) `__print__`
- c) `__repr__`
- d) `__display__`

22. Какой атрибут содержит все методы и поля класса?

- a) `__methods__`
- b) `__dict__`
- c) `__all__`
- d) `__slots__`

23. Какой метод позволяет использовать объект как функцию?

- a) `__func__`
- b) `__call__`
- c) `__run__`
- d) `__execute__`

24. Как динамически добавить метод классу?

- a) `class.add_method()`
- b) Присвоить функцию атрибуту класса
- c) Через `@dynamic`
- d) Никак

25. Как проверить, есть ли атрибут у объекта?

- a) `obj.has_attr()`
- b) `hasattr(obj, 'attr')`
- c) `'attr' in obj`
- d) `obj.attr.exists()`

26. Что такое наследование в Python?

- a) Копирование кода из одного класса в другой
- b) Механизм, позволяющий классу наследовать атрибуты и методы другого класса
- c) Способ объединения нескольких классов в один модуль
- d) Процесс создания экземпляра класса

27. Какой метод вызывается при создании экземпляра дочернего класса?

- a) `__init__` дочернего класса, а если его нет – `__init__` родительского
- b) Только `__init__` родительского класса

- c) `__new__` и `__init__` дочернего класса
- d) Никакой метод не вызывается автоматически

28. Как вызвать метод родительского класса из дочернего?

- a) `Parent.method(self)`
- b) `super().method()`
- c) `self.parent.method()`
- d) `super(Parent, self).method()`

29. Что такое множественное наследование?

- a) Наследование одного класса от нескольких модулей
- b) Наследование одного класса от нескольких родительских классов
- c) Создание нескольких экземпляров одного класса
- d) Наследование от абстрактного класса

30. Как Python обрабатывает порядок наследования при множественном наследовании?

- a) Случайным образом
- b) По алфавиту
- c) С помощью MRO (Method Resolution Order)
- d) По порядку объявления родителей

31. Как можно посмотреть порядок разрешения методов (MRO) класса?

- a) `help(class)`
- b) `class.__mro__` или `class.mro()`
- c) `dir(class)`
- d) `inspect.getmro(class)`

32. Что будет, если два родительских класса имеют метод с одинаковым именем?

- a) Произойдёт ошибка
- b) Будет вызван метод первого родителя в MRO
- c) Будет вызван метод последнего родителя в MRO
- d) Оба метода будут вызваны

33. Что такое переопределение метода?

- a) Удаление метода родительского класса
- b) Создание метода в дочернем классе с тем же именем, что и у родителя
- c) Изменение сигнатуры метода родителя
- d) Копирование метода из другого класса

34. Может ли класс наследоваться от встроенных типов (например, `list`, `dict`)?

- a) Да
- b) Нет
- c) Только если переопределить `__new__`

d) Только в Python 2

35. Как проверить, является ли класс подклассом другого?

- a) `isinstance(obj, Parent)`
- b) `issubclass(Child, Parent)`
- c) `Child in Parent.subclasses()`
- d) `hasattr(Child, Parent)`

36. Можно ли динамически изменить родительский класс объекта?

- a) Да, через `obj.__class__ = NewClass`
- b) Нет, наследование определяется при создании класса
- c) Только через `__bases__`
- d) Только в Python 2

37. Что делает `super().__init__()` в дочернем классе?

- a) Ничего, это синтаксическая ошибка
- b) Вызывает `__init__` родительского класса
- c) Создает новый экземпляр родителя
- d) Удаляет экземпляр родителя

38. Какой из вариантов демонстрирует корректное множественное наследование?

- a) `class Child(Parent1, Parent2):`
- b) `class Child(Parent1)(Parent2):`
- c) `class Child inherits Parent1, Parent2:`
- d) `class Child: Parent1, Parent2`

39. Что произойдет, если в цепочке наследования метод не найден?

- a) Будет вызван метод `object.__missing__`
- b) Возникнет `AttributeError`
- c) Python автоматически создаст пустой метод
- d) Будет вызван метод `__getattr__`

40. Можно ли унаследоваться от класса, определённого в другом модуле?

- a) Да
- b) Нет, только из того же модуля
- c) Только если импортировать его через `import *`
- d) Только если родительский класс — публичный

41. Как запретить добавление новых атрибутов в дочернем классе?

- a) Использовать `@final`
- b) Определить `__slots__` в родительском классе
- c) Сделать все атрибуты приватными
- d) Переопределить `__setattr__`

42. Что выведет `print(Child.__mro__)`, если `Child(Parent1, Parent2)`?

- a) Список всех методов класса
- b) Кортеж с порядком разрешения методов (от `Child` до `object`)
- c) Имена всех родительских классов в случайном порядке

d) None

43. Как проверить, был ли метод переопределён в дочернем классе?

- a) `Child.method == Parent.method`
- b) `Child.method is not Parent.method`
- c) `hasattr(Child, 'overridden')`
- d) `inspect.ismethod(Child.method)`

44. Какой метод позволяет запретить удаление атрибута в дочернем классе?

- a) `__lockattr__`
- b) `__delattr__`
- c) `__slots__`
- d) `__delete__`

45. Можно ли динамически добавить родительский класс?

- a) Да, через `Child.__bases__ += (NewParent,)`
- b) Нет, после создания класса это невозможно
- c) Только через `metaclass`
- d) Только в Python 2

46. Что будет, если в `__init__` дочернего класса не вызвать `super().__init__()`?

- a) Экземпляр не создастся
- b) `__init__` родителя не выполнится, если он не вызван явно
- c) Произойдёт рекурсия
- d) Python вызовет его автоматически

47. Как создать класс, экземпляры которого нельзя изменить?

- a) Сделать все атрибуты приватными
- b) Использовать `__slots__` и не определять `__setattr__`
- c) Наследоваться от `tuple`
- d) Запретить наследование

48. Какой метод вызовется при `obj.method()`, если `method` есть в родителе, но не переопределён?

- a) Метод родительского класса
- b) `AttributeError`
- c) `object.method`
- d) Ничего, синтаксическая ошибка

49. Что будет, если у двух родителей есть атрибут с одним именем?

- a) Ошибка при создании класса
- b) Будет использован атрибут первого родителя в MRO
- c) Будет создан новый атрибут

d) `AttributeError` при обращении

50. Как получить список всех родительских классов?

- a) `dir(Child)`
- b) `Child.__bases__`
- c) `Child.parents()`
- d) `help(Child)`

51. Что такое полиморфизм в ООП?

- a) Возможность изменять код во время выполнения
- b) Способность объектов с одинаковым интерфейсом иметь разную реализацию
- c) Объединение данных и методов в одном классе
- d) Наследование свойств от родительского класса

52. Какой принцип ООП позволяет вызывать один и тот же метод для разных объектов?

- a) Инкапсуляция
- b) Полиморфизм
- c) Наследование
- d) Абстракция

53. Как полиморфизм реализуется в Python?

- a) Только через наследование
- b) Через наследование и перегрузку операторов
- c) Только через абстрактные классы
- d) Через статические методы

54. Можно ли считать полиморфизмом перегрузку операторов?

- a) Нет, это разные концепции
- b) Да, потому что один оператор может работать по-разному для разных классов
- c) Только если используется наследование
- d) Только для числовых типов

55. Как обозначается статический метод в Python?

- a) `@classmethod`
- b) `@staticmethod`
- c) `@property`
- d) `@abstractmethod`

56. Чем отличается статический метод от обычного?

- a) Он не принимает `self` и принадлежит классу, а не экземпляру
- b) Он может изменять состояние класса
- c) Он доступен только из других методов класса

d) Он автоматически вызывается при создании объекта

57. Можно ли обращаться к статическому методу через экземпляр класса?

- a) Нет, только через класс
- b) Да, но это не рекомендуется
- c) Только если метод объявлен как `@classmethod`
- d) Только если класс наследуется от ABC

58. Какой метод перегружает оператор +?

- a) `__add__`
- b) `__plus__`
- c) `__sum__`
- d) `__add__` или `__radd__`

59. Как перегрузить оператор сравнения ==?

- a) `__cmp__`
- b) `__eq__`
- c) `__equals__`
- d) `__compare__`

60. Какой метод перегружает оператор индексирования []?

- a) `__getitem__`
- b) `__setitem__`
- c) `__index__`
- d) `__getitem__` и `__setitem__`

61. Какой декоратор помечает метод как абстрактный?

- a) `@abstract`
- b) `@abstractmethod`
- c) `@virtualmethod`
- d) `@mustoverride`

62. Можно ли создать экземпляр абстрактного класса?

- a) Да, если в нём нет абстрактных методов
- b) Нет, Python выдаст ошибку
- c) Только если класс наследуется от `object`
- d) Да, но только через `@staticmethod`

63. Можно ли сочетать `@staticmethod` и `@abstractmethod`?

- a) Да, без ограничений
- b) Нет, абстрактный метод должен иметь `self`
- c) Только если метод возвращает `None`
- d) Только в классах без наследования

64. Какой из этих методов не является магическим?

- a) `__len__`
- b) `__iter__`

- c) `__static__`
- d) `__contains__`

65. Какой метод перегружает оператор умножения *?

- a) `__mult__`
- b) `__mul__`
- c) `__times__`
- d) `__product__`

66. Как перегрузить оператор +=?

- a) `__plus_equals__`
- b) `__iadd__`
- c) `__aplus__`
- d) `__add_assign__`

67. Как перегрузить оператор in?

- a) `__has__`
- b) `__contains__`
- c) `__in__`
- d) `__include__`

68. Можно ли добавить неабстрактные методы в абстрактный класс?

- a) Нет, только абстрактные
- b) Да, они будут доступны в дочерних классах
- c) Только если класс наследуется от `object`
- d) Только статические методы

69. Какой декоратор делает весь класс абстрактным?

- a) `@abstractclass`
- b) Класс должен наследоваться от `ABC`
- c) `@abstractmethod`
- d) `@force_abstract`

70. Что произойдёт, если в дочернем классе не переопределить все абстрактные методы?

- a) Ошибка при создании экземпляра
- b) Методы останутся пустыми
- c) Класс автоматически станет абстрактным
- d) Ничего, Python проигнорирует это

71. Можно ли обратиться к статическому методу из другого статического метода?

- a) Нет, статические методы изолированы
- b) Да, через имя класса
- c) Только если методы объявлены в одном модуле
- d) Только через `@classmethod`

72. Зачем использовать @staticmethod вместо обычной функции вне класса?

- a) Для логической группировки кода
- b) Чтобы метод был приватным
- c) Для ускорения работы программы
- d) Чтобы запретить наследование

73. Какой принцип ООП не связан напрямую с полиморфизмом?

- a) Инкапсуляция
- b) Наследование
- c) Абстракция
- d) Все связаны

74. Как создать абстрактный класс?

- a) @abstract
- b) from abc import ABC, abstractmethod + наследование от ABC
- c) class Abstract
- d) @abstractclass

75. Что такое "интерфейс" в Python?

- a) Специальный синтаксис для наследования
- b) Абстрактный класс, где все методы — @abstractmethod
- c) Декоратор @interface
- d) Наследование от Interface

76. Какой метод в Pygame вызывается для обновления экрана?

- A) pygame.redraw()
- B) pygame.update_screen()
- C) pygame.display.update()
- D) pygame.screen.refresh()

77. Как загрузить изображение для спрайта в Pygame?

- A) pygame.open_image("sprite.png")
- B) pygame.image.load("sprite.png")
- C) pygame.load("sprite.png")
- D) pygame.sprite("sprite.png")

78. Как создать анимацию персонажа, используя несколько изображений?

- A) Загрузить одно изображение и масштабировать его
- B) Загрузить список изображений и переключать их с задержкой
- C) Использовать только векторную графику
- D) Анимация невозможна в Pygame

79. Какой метод отвечает за обработку столкновений объектов в Pygame?

- A) `check_overlap()`
- B) `detect_collision()`
- C) `colliderect()`
- D) `intersects()`

80. Как сохранить текущий счёт игрока в файл?

- A) `save(score, "score.txt")`
- B) `with open("score.txt", "w") as f: f.write(str(score))`
- C) `file.write(score, "score.txt")`
- D) `export score to "score.txt"`

81. Какой метод в Pygame используется для обработки событий клавиатуры?

- A) `pygame.key_press()`
- B) `pygame.event.get()`
- C) `pygame.input.keyboard()`
- D) `pygame.keyboard.event()`

82. Какой класс в Pygame отвечает за работу с изображениями и спрайтами?

- A) `pygame.Graphics`
- B) `pygame.Draw`
- C) `pygame.Surface`
- D) `pygame.Texture`

83. Какой метод вызывается для отрисовки спрайта на экране?

- A) `pygame.render()`
- B) `screen.blit(sprite, (x, y))`
- C) `pygame.draw_sprite()`
- D) `sprite.display()`

84. Как создать несколько экземпляров врагов в игре?

- A) `enemy = Enemy()`
- B) `enemies = [Enemy() for _ in range(10)]`
- C) Враги не могут создаваться программно
- D) Только вручную через `pygame.add_enemy()`

85. Какой метод используется для проверки столкновения пули и врага?

- A) `bullet.touch(enemy)`
- B) `pygame.sprite.collide_rect(bullet, enemy)`
- C) `if bullet == enemy:`
- D) `bullet.hit_test(enemy)`

86. Как загрузить фоновую музыку в Pygame?

- A) `pygame.load_song("music.mp3")`
- B) `pygame.mixer.music.load("music.mp3")`
- C) `pygame.audio.load("music.mp3")`
- D) `pygame.soundtrack("music.mp3")`

87. Как отображать счет игрока на экране?

- A) `print(score)`
- B) `font.render(f"Score: {score}", True, (255, 255, 255))`
- C) `pygame.show_text(score)`
- D) Только через консоль

88. Как создать меню с кнопками "Start" и "Quit"?

- A) Только через консоль
- B) Отображать прямоугольники (Rect) и проверять клики мыши
- C) `pygame.autocreate_menu()`
- D) Меню нельзя сделать в Pygame

89. Как реализовать систему уровней в игре?

- A) Создать один огромный уровень
- B) Загружать разные карты или параметры при переходе на новый уровень
- C) Уровни нельзя запрограммировать
- D) Только через внешние программы

90. Как реализовать "инвентарь" игрока в ООП-стиле?

- A) Создать глобальный список
- B) Сделать класс Inventory с методами `add_item()`, `remove_item()`
- C) Хранить предметы в переменных `item1`, `item2`...
- D) Инвентарь невозможен в Pygame

91. Как сделать, чтобы игра сохраняла прогресс при закрытии?

- A) Прогресс нельзя сохранить
- B) Записать данные (уровень, HP, инвентарь) в JSON или файл
- C) Использовать `pygame.auto_save()`
- D) Хранить всё в оперативной памяти

92. Как реализовать "подбор предметов" (коллизия игрока и предмета)?

- A) Удалять все предметы на карте
- B) Проверять `colliderect()` и вызывать метод `player.pick_item(item)`
- C) Предметы нельзя подбирать
- D) Использовать `pygame.collect()`

93. Какой тип данных лучше всего подходит для хранения анимаций персонажа?

- A) Строка (str)
- B) Список (list) из Surface-объектов
- C) Число (int)
- D) Словарь (dict) без изображений

94. Как сделать "искусственную задержку" для атаки бота?

- A) Бот должен атаковать каждый кадр
- B) Использовать `pygame.time.get_ticks()` для контроля времени
- C) `time.sleep(1)`
- D) Только через многопоточность

95. Как обработать столкновение пули со стеной?

- A) Пуля проходит сквозь стены
- B) При `colliderect()` удалять пулю из списка
- C) Стены нельзя проверить на коллизии
- D) Использовать `pygame.bullet_stop()`

96. Как реализовать "разные типы урона" (огонь, лед и т.д.)?

- A) Сделать одну переменную `damage`
- B) Создать класс `Damage` с полями `type` и `value`
- C) Типы урона невозможны
- D) Использовать `pygame.damage_type()`

97. Как сделать, чтобы бот "патрулировал" зону?

- A) Бот стоит на месте
- B) Задать точки патрулирования и двигаться между ними
- C) Использовать `pygame.patrol_mode()`
- D) Только через ИИ с ИИ-библиотеками

98. Как реализовать "диалоговые окна" в игре?

- A) Выводить текст в консоль
- B) Создать класс `DialogBox` с отрисовкой текста и кнопок
- C) `pygame.show_dialog()`
- D) Диалоги невозможны

99. Какой метод ООП позволит игроку и боту наследовать общую логику движения?

- A) Композиция
- B) Создать родительский класс `Character` с методом `move()`
- C) Копировать код для каждого объекта
- D) Только через миксины

100. Как оптимизировать игру, если объектов слишком много?

- A) Не оптимизировать
- B) Использовать пространственное разделение (например, квадродерево)
- C) Удалить половину объектов
- D) `pygame.optimize_all()`

ЗАДАЧИ ДЛЯ ПРОВЕРКИ УМЕНИЙ

Компетенция ОПК-3: Способен управлять процессами создания и использования продуктов и услуг в сфере информационно-коммуникационных технологий, в том числе разрабатывать алгоритмы и программы для их практической реализации.

Умения: уметь разрабатывать объектно-ориентированные алгоритмы и программы, пригодные для практического применения.

Задача 1. Класс Person

Создайте класс Person с атрибутами name и age. Добавьте метод introduce(), который выводит: "Меня зовут [name], мне [age] лет."

Задача 2. Класс Rectangle

Создайте класс Rectangle с атрибутами width и height. Добавьте методы: area() — возвращает площадь.
perimeter() — возвращает периметр.

Задача 3. Приватный атрибут в BankAccount

Модифицируйте BankAccount, сделав balance приватным (__balance). Добавьте методы get_balance() и set_balance(amount) (с проверкой: amount ≥ 0).

Задача 4. Класс Temperature с конвертацией

Создайте класс Temperature с приватным атрибутом __celsius. Добавьте: Свойство celsius (геттер/сеттер).
Свойство fahrenheit (геттер/сеттер, конвертация).

Задача 5. Класс Library и Book

Создайте класс Library, который хранит список книг (Book).

Задача 6. Класс Shop и Product

Создайте класс Shop, который может добавлять и продавать товары (Product).

Задача 7. Класс Hospital и Patient

Создайте класс Hospital, который регистрирует пациентов (Patient).

Задача 8. Переопределение __init__

Создайте класс Person с __init__ (имя, возраст) и класс Employee, добавляющий salary.

Задача 9. Наследование от встроенных классов

Создайте класс MyList, наследующий list, добавляющий метод print_all().

Задача 10. Проблема ромбовидного наследования

Создайте иерархию A -> B; A -> C; B, C -> D. Убедитесь, что A.__init__ вызывается 1 раз.

Задача 11. Наследование с добавлением нового метода

Создайте класс `Animal` с методом `eat()` (выводит "Eating..."). Затем создайте класс `Bird`, который наследует `Animal` и добавляет метод `fly()` (выводит "Flying").

Задача 12. Переопределение метода без `super()`

Создайте класс `Vehicle` с методом `move()` ("Moving..."). Затем создайте класс `Car`, который полностью переопределяет `move()` ("Driving a car").

Задача 13. Множественное наследование с общим методом

Создайте классы `A` (метод `show()`: "A") и `B` (метод `show()`: "B"). Затем создайте класс `C(A, B)` и вызовите `show()`. Объясните результат.

Задача 14. Множественное наследование с разными методами

Создайте классы `X` (метод `a()`: "X.a") и `Y` (метод `b()`: "Y.b"). Затем создайте класс `Z(X, Y)` и вызовите оба метода.

Задача 15. Животные и их звуки

Создайте классы `Dog` и `Cat` с методом `make_sound()`. Используйте полиморфизм, чтобы вызвать этот метод для объектов разных классов.

Задача 16. Фигуры с методом `draw()`

Создайте классы `Circle` и `Square` с методом `draw()`, который возвращает строку-описание фигуры. Продемонстрируйте полиморфизм.

Задача 17. Калькулятор

Создайте класс `Calculator` со статическими методами для сложения, вычитания и умножения.

Задача 18. Вектор с перегрузкой +

Создайте класс `Vector` с перегруженным оператором `+` для сложения векторов.

Задача 19. Фигуры с методом `area()`

Создайте абстрактный класс `Shape` с методом `area()`. Реализуйте его в `Circle` и `Rectangle`.

Задача 20. Абстрактный транспорт

Создайте абстрактный класс `Transport` с методом `move()`. Реализуйте его в `Car` и `Boat`.

Задача 21. Банковский счёт с перегрузкой операторов

Создайте класс `BankAccount` с перегрузкой `+` (пополнение) и `-` (снятие), а также статическим методом для проверки суммы.

Задача 22. Создание класса игрока

Создайте класс `Player`, который имеет:

Атрибуты: `x`, `y`, `speed`, `image` (загруженное изображение).

Метод `draw()`, отрисовывающий игрока на экране.

Задача 23. Движение игрока по клавишам

Добавьте в класс `Player` метод `update()`, который перемещает объект по нажатию WASD.

Задача 24. Загрузка спрайта вместо квадрата

Замените `pygame.Surface` на загрузку изображения из файла `player.png`.

Задача 25. Создание класса врага

Создайте класс `Enemy`, который движется к игроку.

Задача 26. Столкновения игрока и врага

Добавьте проверку коллизии между `Player` и `Enemy`. При столкновении выводите "Game Over".

Задача 27. Анимация бега игрока

Загрузите 4 кадра анимации и переключайте их при движении.

Задача 28. Создание пуль

При нажатии SPACE создавайте пулю, которая летит вперед.

ЗАДАЧИ ДЛЯ ПРОВЕРКИ НАВЫКОВ

Компетенция ОПК-3: Способен управлять процессами создания и использования продуктов и услуг в сфере информационно-коммуникационных технологий, в том числе разрабатывать алгоритмы и программы для их практической реализации.

Навыки: владеть навыками разработки объектно-ориентированных алгоритмов и программ, пригодных для практического применения.

Задача 1. Класс `BankAccount`

Создайте класс `BankAccount` с атрибутом `balance` (начальный баланс = 0).

Реализуйте методы:

`deposit(amount)` — пополнение счёта.

`withdraw(amount)` — снятие денег (если средств достаточно).

Задача 2. Класс `Student`

Создайте класс `Student` с атрибутами `name` и `grades` (список оценок). Добавьте метод `average_grade()`, возвращающий средний балл.

Задача 3. Класс Car

Создайте класс Car с атрибутами brand, model, year. Добавьте метод get_info(), возвращающий строку с данными.

Задача 4. Класс Password с валидацией

Создайте класс Password с приватным атрибутом __password. Реализуйте: Метод set_password(pwd) (только если длина ≥ 8 символов). Метод check_password(pwd) (проверяет соответствие).

Задача 5. Класс Person с валидацией возраста

Создайте класс Person с приватным атрибутом __age (только ≥ 0). Добавьте свойство age с геттером/сеттером.

Задача 6. Класс Student с защищёнными оценками

Создайте класс Student с приватным списком __grades. Добавьте: Метод add_grade(grade) (добавляет оценку 1-10). Метод get_grades() (возвращает копию списка).

Задача 7. Класс University и Student

Создайте класс University, который управляет студентами (Student).

Задача 8. Класс Playlist и Song

Создайте класс Playlist, который хранит песни (Song).

Задача 9. Наследование и добавление атрибута

Создайте класс Book с __init__ (принимает title). Затем создайте класс EBook, который добавляет атрибут format.

Задача 10. Наследование и вызов метода из родителя

Создайте класс Alpha с методом run() ("Alpha runs"). Затем создайте класс Beta, который вызывает run() из Alpha.

Задача 11. Наследование и изменение списка

Создайте класс Parent с атрибутом items = [1, 2]. Затем создайте класс Child, который добавляет 3 в items.

Задача 12. Иерархия классов с многоуровневым наследованием

Создайте цепочку наследования:

A (метод test(): "A")

B(A) (переопределяет test(): "B")

C(B) (переопределяет test(): "C")

D(C) (вызывает test() всех родителей через super()).

Задача 13. Множественное наследование с пересекающимися методами

Создайте классы:

X (метод process(): "X.process")

Y (метод process(): "Y.process")

Z(X, Y) (вызывает process() через super()).

Задача 14. Множественное наследование с инициализацией

Создайте классы:

Alpha (__init__ принимает x, сохраняет в self.x)

Beta (__init__ принимает y, сохраняет в self.y)

Gamma(Alpha, Beta) (инициализирует оба родителя).

Задача 15. Наследование и вызов методов из разных родителей

Создайте классы:

A (метод start(): "A.start")

B (метод stop(): "B.stop")

C(A, B) (метод restart() вызывает stop(), затем start()).

Задача 16. Множественное наследование с общим предком

Создайте иерархию:

Animal (метод sound(): "Какой-то звук")

Dog(Animal) (переопределяет sound(): "Гав")

Cat(Animal) (переопределяет sound(): "Мяу")

Pet(Dog, Cat) (вызывает sound() через super()).

Задача 17. Игровые персонажи

Создайте классы Warrior и Mage с методом attack(), возвращающим разный урон (10 и 15 соответственно).

Задача 18. Валидация email

Добавьте в класс User статический метод validate_email(), который проверяет, содержит ли email символ @.

Задача 19. Кеширование данных

Создайте класс Cache со статическим словарём для хранения данных и методами set(key, value) и get(key).

Задача 20. Перегрузка == для класса Book

Перегрузите оператор == для класса Book, чтобы книги сравнивались по названию и автору.

Задача 21. Перегрузка * для матриц

Создайте класс Matrix с перегруженным оператором * для умножения матриц.

Задача 22. Абстрактный класс Animal

Создайте абстрактный класс Animal с методами eat() и sleep(). Реализуйте его в Lion и Fish.

Задача 23. Генератор случайных чисел с полиморфизмом

Создайте классы RandomInt и RandomFloat с методом generate(), возвращающим случайное число. Используйте полиморфизм.

Задача 24. Абстрактный класс Device с перегрузкой ==

Создайте абстрактный класс Device с методом turn_on() и перегрузите == для сравнения по мощности. Реализуйте в Lamp и TV.

Задача 25. Удаление пуля за пределами экрана

Если пуля вышла за границы экрана, удалите её из списка.

Задача 26. Система здоровья (HP)

Добавьте игроку hp = 100. При столкновении с врагом отнимайте 10 HP.

Задача 27. Сохранение рекорда в файл

При завершении игры сохраняйте счет в score.txt.

Задача 28. Случайное движение врагов

Реализуйте движение врагов случайным образом с изменением направления каждые 2 секунды.

Задача 29. Меню паузы

При нажатии P ставьте игру на паузу с выводом текста "Paused".

Задача 30. Подбор предметов (монет)

Создайте класс Coin. При столкновении с игроком увеличивайте счет.

Задача 31. Параллакс-фон

Реализуйте фон с 2 слоями, движущимися с разной скоростью.

Задача 32. Диалоговая система с NPC

Создайте класс NPC, который:

При приближении игрока выводит текст диалога.

Диалог отображается в прямоугольнике на экране.

Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Система критериев оценки определяет оценку успеваемости по каждому заданию (вопросу) экзаменационного билета или заданию для зачета с использованием интервальной шкалы баллов, применяемой в привязке к рейтинговой 100-балльной системе.

ОЦЕНКА ОТВЕТА НА ТЕОРЕТИЧЕСКИЙ ВОПРОС В УСТНОЙ ИЛИ ПИСЬМЕННОЙ ФОРМЕ:

Оценка «отлично» / «зачтено» (91-100 баллов) выставляется при соблюдении следующих условий: Ответ отличается глубиной и полнотой, свободным владением понятийно-категориальным (терминологическим) аппаратом изученной дисциплины. Отражает знание не только основной, но и дополнительной литературы. Приведены примеры, отражающие умение связать теорию с практикой. Ответ изложен логически последовательно, грамотно и корректно.

Оценка «хорошо» / «зачтено» (76-90 баллов) выставляется при соблюдении следующих условий: Ответ отличается полнотой, владением понятийно-категориальным (терминологическим) аппаратом изученной дисциплины, но в ответе могут присутствовать неточности. Отражает знание основной литературы. Приведены примеры, отражающие умение связать теорию с практикой. Ответ изложен логически последовательно, грамотно и корректно, но недостаточно аргументирован.

Оценка «удовлетворительно» / «зачтено» (61-75 баллов) выставляется при соблюдении следующих условий: В ответе отражено знание понятийно-категориального (терминологического) аппарата изучаемой дисциплины, но присутствуют отдельные ошибки и неточности. Ответ характеризуется недостаточным знанием рекомендованной литературы. Примеры, отражающие умение связать теорию с практикой, тривиальны, либо отсутствуют. Ответ неполный, носит фрагментарный, непоследовательный характер.

Оценка «неудовлетворительно» / «не зачтено» (0-60 баллов) выставляется при соблюдении следующих условий: Ответ характеризуется незнанием, либо фрагментарным представлением о понятийно-категориальном аппарате дисциплины, содержит множество ошибок. Примеры и иллюстрации отсутствуют. Ответ логически непоследователен.

ОЦЕНКА ВЫПОЛНЕНИЯ ЗАДАНИЯ В ФОРМЕ CASE-STUDY (СИТУАЦИИ)

Оценка «отлично» / «зачтено» (91-100 баллов) выставляется при соблюдении следующих условий: Четкая формулировка проблемы. Полное и соответствующее ситуации решение, основанное на знании правовых норм и технологий (опыте), применяемых в реальных организациях (известных компаниях). Предполагаемые действия описаны логично и последовательно.

Даны дополнительные авторские комментарии и предложения к решению ситуации.

Оценка «хорошо» / «зачтено» (76-90 баллов) выставляется при соблюдении следующих условий: Понимание сути проблемы, но ее формулирование затруднено. Решение соответствует ситуации, отражает знание правовых норм и опыт работы других организаций при решении подобных ситуаций. Логика и последовательность действий не нарушены.

Оценка «удовлетворительно» / «зачтено» (61-75 баллов) выставляется при соблюдении следующих условий: Проблема не сформулирована. Приведен набор действий, потенциально способствующих улучшению ситуации и решению проблемы.

Оценка «неудовлетворительно» / «не зачтено» (0-60 баллов) выставляется при соблюдении следующих условий: Предложенный перечень мероприятий не соответствует ситуации.

ОЦЕНКА РЕШЕНИЯ ЗАДАЧИ

Оценка «отлично» / «зачтено» (91-100 баллов) выставляется при соблюдении следующих условий: Полное верное решение - оценивается в n баллов (n – максимальное количество баллов за решение задачи в структуре экзаменационного билета/задания).

Оценка «хорошо» / «зачтено» (76-90 баллов) выставляется при соблюдении следующих условий: Верное решение; имеются небольшие недочеты, в целом не влияющие на решение – оценивается в диапазоне от $0,76*n$ баллов до $0,9*n$ баллов (n – максимальное количество баллов за решение задачи в структуре экзаменационного билета/задания).

Оценка «удовлетворительно» / «зачтено» (61-75 баллов) выставляется при соблюдении следующих условий: Решение в целом верное; однако оно содержит ряд ошибок, либо не учитывает отдельных случаев, но может стать правильным после некоторых исправлений или дополнений – оценивается в диапазоне от $0,61*n$ баллов до $0,75*n$ баллов (n – максимальное количество баллов за решение задачи в структуре экзаменационного билета/задания).

Оценка «неудовлетворительно» / «не зачтено» (0-60 баллов) выставляется при соблюдении следующих условий: Решение неверное; изначально выбран неверный ход решения, или решение отсутствует – оценивается в 0 баллов.

ОЦЕНКА ВЫПОЛНЕНИЯ ТЕСТОВОГО ЗАДАНИЯ

Подсчитывается доля набранных баллов в максимальной сумме баллов за все задания теста:

– Каждый правильный ответ на тестовый вопрос (тип выборочный, одинарный, множественный, открытый) оценивается в m баллов (число m определяется путем деления максимального количества баллов за выполнение теста в структуре экзаменационного билета/задания на количество тестовых заданий);

– Каждый частично правильный ответ на тестовый вопрос (тип выборочный, множественный, открытый) оценивается в $m/2$ баллов независимо от соотношения правильно/неправильно выбранных вариантов (число m определяется путем деления максимального количества баллов за выполнение теста в структуре экзаменационного билета/задания на количество тестовых заданий);

– Каждый неправильный ответ на тестовый вопрос (тип выборочный, одинарный) оценивается в 0 баллов.

Оценка «отлично»/ «зачтено» (91-100 баллов) выставляется, если доля набранных баллов составляет 91-100%.

Оценка «хорошо»/ «зачтено» (76-90 баллов), если доля набранных баллов составляет 76-90%.

Оценка «удовлетворительно»/ «зачтено» (61-75 баллов), если доля набранных баллов составляет 61-75%.

Оценка «неудовлетворительно»/ «не зачтено» (0-60 баллов), если доля набранных баллов составляет не более 60%.