Классы и объекты в Java. Принципы ООП

Рассматриваемые вопросы

- Основы системного подхода
- Понятие предметной области (ПрО). Программа как модель ПрО
- Классы в Java:
  - описание (декларация) и определение (реализация)
  - создание и использование экземпляров класса
- Различие понятий «класс» и «объект»
- Модификаторы доступа класса и его элементов
- Конструкторы класса
- Вызов методов и передача им параметров
- Основные принципы ООП. Их реализация в Java
  - инкапсуляция
  - наследование
  - полиморфизм

Основы системного подхода (СП)

- Системный подход — это метод познания мира, в его основе — рассмотрение объектов в качестве систем
- Основные допущения СП:
  - В мире существуют системы
  - Системное описание истинно
  - Системы взаимодействуют друг с другом, а, следовательно, всё в этом мире взаимосвязано
  - Следовательно мир — это тоже система

Основные понятия СП

- Система — совокупность взаимосвязанных элементов, образующих целостность или единство
- Структура — способ взаимодействия элементов системы посредством определенных связей («картина связей»)
- Состояние — положение системы относительно других её положений в пространстве состояний
- Процесс — динамическое изменение системы во времени (переход из одного состояния в другое).

Предметная область (ПрО)

- ПрО – это множество всех предметов, свойства кот. и отношения м-ду кот. рассматриваются.
- Любая программа является моделью (образом, представлением) ПрО
- Реальные ПрО достаточно сложны, состояние их составляющих элементов описывается с помощью большого набора параметров
- Оно не может быть описано с помощью единственного примитивного типа данных
- Для решения проблемы сложности были созданы классы – элементы более высокого уровня

Примеры ПрО

- ПрО: **Банк**
- Составные элементы: клиент (вкладчик, кредитор и т.п), счёт, транзакция (перевод денег) и т.п.
- ПрО: **компьютер**
- Элементы: комплектующие компьютера
- ПрО: **человек**
- Элементы: органы и подсистемы человека
- ПрО: **коммерческое предприятие**
- Элементы: покупатель, поставщик, товар, группа товаров, продажа и т.п.

Классы

- Класс – это также тип данных, но более сложный (составной, состоящий из субэлементов)
- Класс – это описание объектов, которые будут создаваться на основе этого описания
- Аналогия: класс – это проект (чертёж) типового дома, а объект – это конкретный дом, кот.будет построен по этому проекту
- Все дома, построенные по типовому проекту, будут иметь индивидуальные особенности (например, адрес)

Последовательность действий при работе с классами

1. Описать класс
2. Реализовать класс
3. Создать экземпляр класса – объект
4. Работать с объектом, обращаясь к его свойствам и методам
5. Уничтожить объект (в Java – автоматически)

Описание класса

- В отличие от C++, в Java описание (объявление, declaration) класса совмещено с его определением (definition)
- Т.е. методы класса нужно реализовывать (писать код тела) сразу же при их описании
- Как правило, в Java класс описывается в отдельном файле.
- Синтаксис:

```
[Модификатор доступа] class ИмяКласса {
//члены класса – свойства и методы
  Модиф._доступа Тип имяСвойства [=значение];
  Модиф._доступа Тип имяМетода(параметры) {
    //тело метода
   }
}
```

Пример описания класса

```
// общедоступный класс Автомобиль
public class Car {
    public int maxSpeed; //макс. скорость
    public int currentSpeed; //текущая скорость
    public String vendor; //производитель
    public String model; // модель

    // закрытая константа «шаг приращения скорости»
    private final int speedOnSteep = 5;

    // открытый метод «газонуть»
    public void stepOn() {
        if (currentSpeed + speedOnSteep <= maxSpeed) {
            currentSpeed += speedOnSteep;
        }
    }

    // вывод информации об объекте
    public void showDescription() {
        System.out.println("Vendor: " + vendor +
        "\nModel: " + model + "\nCurrentSpeed: " +
currentSpeed);
    }
}
```

Создание объектов

- Тот факт, что в программе описан класс, ещё не означает, что мы можем работать с его свойствами и вызывать его методы
- Для работы требуется: 1) объявить переменную типа *Класс* и 2) создать объект (экземпляр) класса
- Синтаксис:

```
ИмяКласса имяОбъекта; //объявление переменной-объекта
имяОбъекта = new ИмяКласса([параметры конструктора]);
```

- Можно совместить объявление и создание объекта:

```
ИмяКласса имяОбъекта = new ИмяКласса([параметры
конструктора]);
```

- Например:

```
Car kalina;
kalina = new Car();
Car merz = new Car();
```

объектов

1. Создаться ссылочная переменная в стеке для хранения адреса будущего объекта
2. В *куче* (heap) выделяется пространство для размещения объекта со всеми его свойствами (атрибутами)
3. Атрибуты инициализируются значениями по умолчанию
4. Выполняется явная инициализация атрибутов, если она была задана программистом
5. Выполняется конструктор
6. Ссылка на созданный объект (его адрес) записывается в соответствующую ссылочную переменную

Использование объекта

- Использование объекта осуществляется посредством доступа к его элементам
- Для доступа к элементам используется оператор «точка» после переменной-ссылки на объект:

```
имяОбъекта.имяСвойства = значение;
имяОбъекта.имяМетода(параметры);
```

- Например:

```
kalina.vendor = "ВАЗ";
kalina.model = "Калина";
kalina.maxSpeed = 160;
kalina.showDescription();

merz.vendor = "Mercedes";
merz.model = "S500";
merz.maxSpeed = 220;
```

Атрибуты класса

- Атрибуты (свойства) класса – это переменные внутри класса
- Совокупность значений атрибутов объекта описывает состояние этого объекта
- Атрибуты класса, в отличие от локальных переменных, инициализируются значениями по умолчанию
  - Числовые элементы – нулями
  - Символьные – значением ‘\0’ (нулевой символ)
  - Логические – значением false
  - Ссылки на объекты – значениями **null**
- Атрибуты класса могут быть инициализированы явным образом при их объявлении:
  - public int age = 0;

Методы класса

- Метод – это функция, описанная внутри класса
- Совокупность методов определяет поведение класса
- Описание метода включает заголовок и тело:
- [модификаторы] тип имя(параметры) { тело; }
- Тело – совокупность операторов
- Например:
- void printHello() { System.out.println(“hello”); }
- Метод может принимать параметры и возвращать значение:
- int square(int x) { return x*x; }

Методы класса

- Если метод возвращает значение простого типа или ссылку на объект, то его тип должен быть указан в заголовке метода
- Также в теле метода должен содержаться хотя бы один оператор **return**
- Если метод не возвращает значение, то в его заголовке должен быть указан тип void
- Оператор return немедленно прекращает выполнение метода и возвращает управление вызывающему методу
- Хороший стиль – использование одного оператора return в одном методе
- Но Java не запрещает многократное использование return (при наличии соответствующих условий)

Вызов методов с параметрами

- Формальные параметры описаны в заголовке метода
- Фактические параметры указываются при его вызове
- Если в заголовке метода описаны *формальные* параметры, то при его вызове нужно указывать фактические параметры
- Значение, переданные в качестве фактических параметров, копируются в переменные-формальные параметры
- В качестве фактических параметров могут выступать константы, переменные или выражения требуемого типа или типа, приводимого к нему неявно

```
int k = 12;
double m;
. . .
m = mult( 23.7, k );
```

```
public double mult( double p_multiplier1, double p_multiplier2 )
{
   return p_multiplier1 * p_multiplier2;
}
```

Передача параметров в методы

- В языке Java при вызове методов передача значений фактических параметров в формальные параметры осуществляется копированием.
- Изменение значения формального параметра не влияет на значение фактического параметра.
- В этой связи утверждается, что значение из фактических параметров в формальные осуществляется по значению.
- При передаче ссылок на объекты осуществляется копирование фактического параметра, то есть *копируется ссылка*. После такого копирования, и фактический параметр, и формальный параметр, ссылаются на один и тот же объект.
- Напоминание: все объекты в Java являются ссылками

Конструктор класса

- Конструктор класса – это специальный метод, название которого совпадает с именем класса
- Конструктор вызывается автоматически при создании объекта
- Конструктор не может возвращать значение
- Каждый класс обязан иметь конструктор.
- Если в классе никакого конструктора явно не написано, то система автоматически создает конструктор без параметров, который называется конструктором по умолчанию.
- Если в классе явно описан какой-либо конструктор, то конструктор по умолчанию системой не создается .
- Конструкторы также, как и другие методы, может иметь модификатор доступа
- Один класс может иметь несколько конструкторов с разными параметрами (перегрузка (overload) конструктора)

Применения конструкторов

- С помощью конструкторов можно управлять процессом создания объекта и производить какие-то действия, обычно связанные с первичной настройкой (инициализацией) объекта:
  - присвоение атрибутам значений по умолчанию
  - соединение с БД
  - соединение с сетью
  - создание других объектов
  - и др.

класса

```
// общедоступный класс Автомобиль
public class Car {
    public int maxSpeed; //макс. скорость
    public int currentSpeed; //текущая скорость
    public String vendor; //производитель
    public String model; // модель

    // закрытая константа «шаг приращения скорости»
    private final int speedOnSteep = 5;

    // конструктор класса
    public Car(int maxSp, String vend, String model) {
        maxSpeed = maxSp;
        vendor = vend;
        this.model = model; //используем ссылку this на текущий объект
    }

    // другие методы ….
}

// Car c = new Car(); Ошибка! Конструктора по умолчанию без параметров нет
Car kalina = new Car(160, "ВАЗ", "Калина");
```

Уничтожение объекта

- В Java нет деструкторов класса
- Уничтожение неиспользуемых объектов осуществляется автоматически «сборщиком мусора» (garbage collector), специальным механизмом JVM
- Объект удаляется, когда в последующей программе на него нет ни одного обращения

Модификаторы доступа (МД)

- Каждый элемент класса должен иметь свой МД
- МД определяет «видимость» этого элемента для других классов
- Весь класс также имеет свой МД

Модификаторы доступа

Следующая таблица, описывает области видимости для разных компонентов классов с разными модификаторами доступа.

Модификатор доступа	**Тот же самый класс**	**Тот же самый пакет**	**Субкласс (класс потомок, возможно из другого пакета)**	**Внешние классы из других пакетов**
`private`	Да			
модификатор доступа опущен – пакетный доступ (default)	Да	Да		
`protected`	Да	Да	Да	
`public`	Да	Да	Да	Да

Специальная ссылка this

- this – это ссылка на текущий экземпляр объекта
- Может применяться только в методах класса
- С помощью this можно:
  - обращаться к атрибутам текущего объекта
  - вызывать методы текущего объекта
  - передавать ссылку на текущий объект методам другого класса

Задание

- По аналогии с созданным классом Rectangle создать в отдельных файлах классы Triangle (треугольник) и Circle (круг) – описать их свойства (атрибуты) и реализовать методы (включая рисование).