Лабораторная 2

1. Объектно-ориентированное программирование. Основные понятия: объекты, наследование, полиморфизм, инкапсуляция.

Объект(экземпляр класса)

Это программная модель реальных объектов или абстрактным понятий, которая представляет собой совокупность переменных, задающих состояние объекта, и связанных сними методов, определяющих поведение объекта.

Наследование

Это приобретение одним классом(подклассом) свойств другого класса(суперкласса). Поля и методы доступны в подклассе при наследовании (если есть модификаторы protected и public, то они доступны). В подклассе могут быть ещё определены свои методы и поля.

Полиморфизм

> *Полиморфизм* — возможность единообразно обрабатывать объекты разных типов. Различают *полиморфизм включения* (или полиморфизм подтипов), который позволяет обращаться с помощью единого интерфейса к классу и к любым его потомкам, и *параметрический полиморфизм*, который определяет обобщенный интерфейс с одинаковой реализацией.

Объявления классов	***Использование полиморфизма***

`public class Shape {`
`  abstract void draw();`
`}`
 	
`public class Square extends Shape {`
`  void draw() {`
`  ...`
`  }`
`}`
 	
`public class Circle extends Shape {`
`  void draw() {`
`  ...`
`  }`
`}`
	`public class FiguresSet {`
	`  public static`
	`  void main(String[] args) {`
	`  ...`
	`  Shape square;`
	`  Shape circle;`
	`  ...`
	`  square = new Square();`
	`  circle = new Circle();`
	`  ...`
	`  square.draw();`
	`  circle.draw();`
	`  }`
	`}`

Конструктор

Это специальный метод, который имеет имя, совпадающее с именем класса, и вызывается при создании экземпляра объекта совместно с оператором **new**. Результатом работы этого метода всегда является экземпляр класса.(**конструктор используется для инициализации объекта**)

Конструкторы имеются у всех классов, поскольку Java автоматически предоставляет конструктор, используемый по умолчанию (без параметров) и инициализирующий все переменные экземпляра их значениями, заданными по умолчанию.

`this()` означает ссылку на текущий объект(экземпляр класса).
`super()` - текущий экземпляр родительского класса.

И `this`, и `super` могут использоваться внутри конструкторов для вызова других конструкторов по цепочке, например, `this()` и `super()` вызывают конструктор без аргументов наследующего и родительского классов соответственно

Инкапсуляция

Является механизмом обёртывания данных (переменных) и кода, работающего с данными (методами), в одно целое. В инкапсуляции переменные класса будут скрыты от других классов и доступ к ним может быть получен только с помощью метода их текущего класса.

Объявление класса	***Доступ к переменным класса***

`public class Point {`
`  private int x, y;`
 	
`  public void move(x1,y1) {`
`  x = x1;`
`  y = y1;`
`  }`
 	
`  public int getX() { return x; }`
`  public int getY() { return y; }`
`}`
	`Point a = new Point(20,30);`
	`int z;`
	
	~~`a.x = 25;`~~ `// запрещено`
	~~`z = a.y;`~~ `// запрещено`
	
	`z = a.getY();`
	`a.move(25,a.getY());`
	
	`a.move(a.getX()-5,a.getY()+5);`

2. Понятие класса. Классы и объекты в Java.

Поле - это переменная, которая описывает какое-либо из свойств данного класса

Класс - это описание еще не созданного объекта, как бы общий шаблон, состоящий из полей, методов и конструктора.

Объявление класса	***Реализация объекта***

`class Point {`
`  int x, y;`
`  boolean isVisible;`
`}`
	`Point a; // объявление объекта`
	`a = new Point(); // создание объекта`
	`a.x = 40;`
	`a.y = 25;`
	`a.isVisible = true;`

Класс может иметь несколько различных конструкторов. Они как и методы отличаются между собой количеством, типом и порядком следования параметров. Если в классе несколько конструкторов с разным набором параметров, это называется *перегрузкой конструктора*.

3. Члены класса. Модификаторы доступа.

public- метод или переменная доступна везде, где доступен класс, в котором она объявлена

protected - метод или переменная доступны в подклассах и в пределах пакета

private - метод или переменная доступны только в пределах класса

default

1) Используется для обеспечения уровня инкапсуляции внутри пакета.
2) Подходит для классов и методов, которые должны быть доступны **только внутри одного пакета** и **скрыты от классов из других пакетов**
3) Применяется по умолчанию в случае, если модификатор доступа не указан.

4. Создание и инициализация объектов. Вызов методов.

Cat puma = new Cat()

В первой части происходит объявление ссылки на объект. Во второй части создается физическая копия объекта, ссылка на него присваивается переменной puma, это делается с помощью оператора new. Оператор new выделяет память для объекта и возвращает ссылку на него.

> Объекты взаимодействуют между собой путем обмена *сообщениями*, которые могут содержать *параметры*. Отправка объекту сообщения осуществляется с помощью вызова соответствующего метода этого объекта.

Объявление класса	*Вызов методов*

class Point {
  int x, y;
  boolean isVisible;
 	
  void hide() {
  isVisible = false;
  }
 	
  void show() {
  isVisible = true;
  }
  void move(x1, y1) {
  x = x1;
  y = y1;
  }
}
	Point a = new Point(20,30);
	a.isVisible = true;
	
	a.hide();
	a.move(60,80);
	a.show();
	
	Point center = new Point();
	center.show();

5. Области видимости переменных.

Область видимости переменной начинается с момента её объявления.

Локальные переменные объявляются внутри методов, конструкций `if`, `for`, `while` или других блоков кода. Они доступны только внутри этого блока и недоступны за его пределами.

Переменные экземпляра объявляются внутри класса. Доступны во всех методах и блоках внутри этого класса, но доступны только через конкретный объект.

Статические переменные объявляются с модификатором `static` и принадлежат классу, а не конкретному экземпляру. Доступна во всех методах класса. Доступна для других классов, если имеет соответствующую область видимости.

Аргументы метода находятся только в теле метода, в котором они объявлены.

6. Модификаторы final и static

static - Когда переменная объявляется статической, это означает, что переменная принадлежит именно классу, а не какому-либо конкретному экземпляру класса. То есть в памяти находится только одна копия переменной, независимо от того, сколько экземпляров класса было создано.

final - переменная является константой, метод не может быть переопределен, класс не может быть расширен

7. Пакеты, инструкция import

Пакет (Package) в Java - это способ объединить группу классов, интерфейсов и подпакетов. Пакеты создаются с помощью ключевого слова `package`, которое указывается в начале файла.

Инструкция `import` используется для того, чтобы включить классы или целые пакеты из других пакетов, чтобы можно было обращаться к ним напрямую, не указывая полный путь.

Дополнительная информация

Интерфейс

В интерфейсах можно описать и записать поля, методы, которые должны реализоваться во всех классах, использующих интерфейс

> *Интерфейс* — абстрактное описание набора методов и констант, необходимых для реализации определенной функции.

Объявление интерфейса	***Реализация интерфейса***

`interface Displayable {`
`  void hide();`
`  void show();`
`}`
	`public class Pixel extends Point`
	`  implements Displayable {`
	
	`  ...`
	`  void hide() {`
	`  ...`
	`  }`
	`  void show() {`
	`  ...`
	`  }`
	`}`

Методы в интерфейсах. В интерфейсах можно определять **три типа методов**:

1. **Абстрактные методы** - это методы, которые не имеют тела (реализации) в интерфейсе. Любой класс, который реализует интерфейс, обязан предоставить реализацию для этих методов.
2. **Методы с реализацией по умолчанию** (default методы) - методы с реализацией, которые класс может переопределить, но не обязан.
3. **Статические методы** - методы, которые принадлежат самому интерфейсу и не могут быть переопределены в классе.

Поля в интерфейсах всегда `public static final` (они являются константами)

Вложенные, анонимные и локальные классы

Вложенный класс — это класс, объявленный внутри объявления другого класса.
Локальный класс — это класс, объявленный внутри блока кода. Область видимости локального класса ограничена тем блоком, внутри которого он объявлен.

```
public class EnclosingClass {
  public void enclosingMethod(){
    // Этот класс доступен только внутри enclosingMethod()
    public class LocalClass {
      ...
    }
  }
}
```

Анонимный класс - локальный класс без имени

```
public class EnclosingClass {
  public void enclosingMethod(){
    // Параметр конструктора – экземпляр анонимного класса,
    // реализующего интерфейс Runnable
    new Thread(new Runnable() {
      public void run() {
        ...
      }
    }).start();
  }
}
```

Аннотации

Аннотации — это модификаторы, семантически не влияющие на программу, но содержащие метаданные, которые могут быть использованы при анализе кода, в процессе компиляции программы или во время её исполнения.

@Override - Переопределяет метод родительского класса

@SupressWarnings - блокирует предупреждение компилятора

CLASSPATH

CLASSPATH – это переменная окружения, которая сообщает JVM, где искать пользовательские и системные классы. Это может быть путь к директории, где хранятся скомпилированные классы, или путь к jar-файлам
Пример:
export CLASSPATH=./lib/Pokemon.jar:./src:./src/info/attacks/physical

moves:./src/info/attacks/statusmoves:./src/info/attacks/specialmoves:./src/info/pokemons:./src/info/Main

`./lib/Pokemon.jar` - внешняя библиотека.

`./src` - путь к корневой папке исходного кода программы (`src`)

Дальше указываем пути к пакетам с различными классами

Создание jar-файла с помощью MANIFEST.MF

После установки CLASSPATH

1) javac -d out src/info/Main.java

Эта команда компилирует исходный файл `Main.java`, который находится в `src/info/`. Флаг `-d out` указывает, что скомпилированные `.class` файлы должны быть помещены в директорию `out`.

2) echo -e "Manifest-Version: 1.0\nClass-Path: lib/Pokemon.jar\nMain-Class: info.Main\n" > MANIFEST.MF

Указываем версию манифеста, путь к нашей внешней библиотеки(JAR-файлу). `Main-Class: info.Main` — указывает основной класс, содержащий метод `main()`, который будет запущен при запуске JAR-файла. Записываем текст в файл MANIFEST.MF

3) jar -cfm Main.jar MANIFEST.MF -C out .

cfm - флаги, указывающие на создание нового JAR-файла (`c`), использование файла манифеста (`m`), и добавление в JAR-файл содержимого.

`-C out .` — добавляет содержимое директории `out` (скомпилированные классы) в корень JAR-файла