



Модуль 2. Введение в объектно-ориентированное программирование

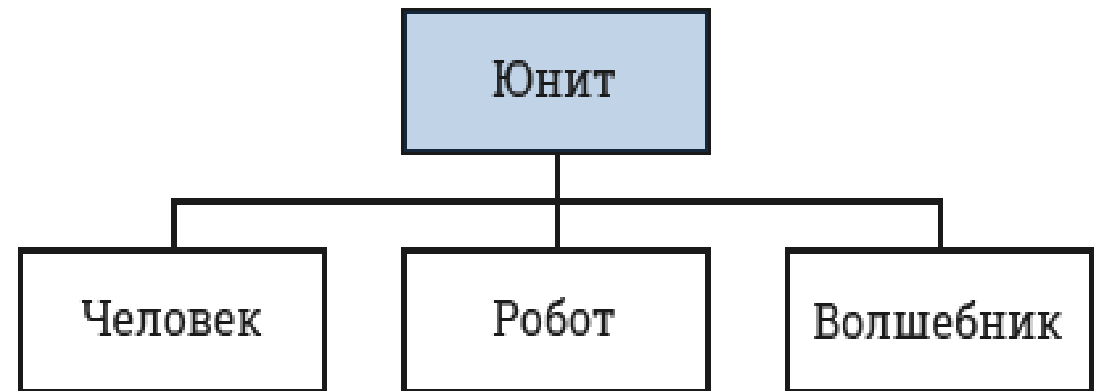
2.6. Наследование и полиморфизм в Java

Понятие наследования

○○○ ○○○

Наследование — это отношение между классами, при котором класс использует структуру или поведение другого класса.

```
class Unit{  
    private String name;  
    private int health;  
  
    public Unit(String name, int health) {  
        this.name = name;  
        this.health = health;  
    }  
  
    public void printInfo(){  
        System.out.println("Name:" + name);  
        System.out.println("Health:" + health);  
    }  
}
```



Понятие наследования

○○○ ○○○

```
public class Robot extends Unit{ }
```

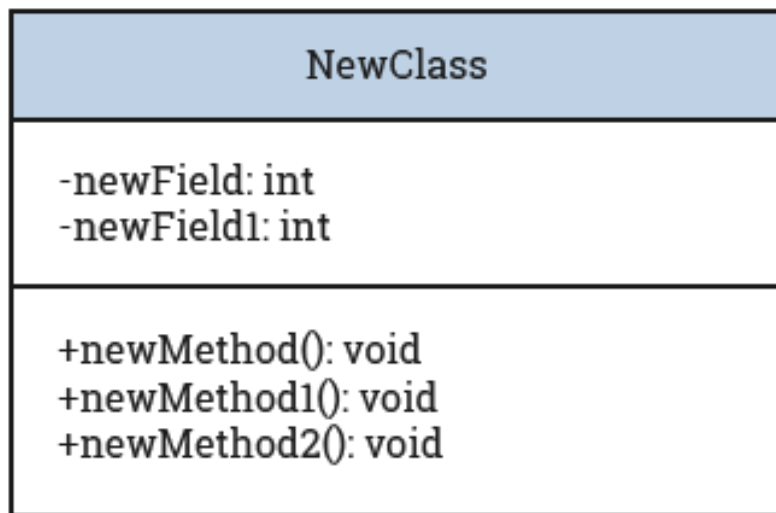
```
public class MainActivity extends Activity { }
```

```
class Robot extends Unit{  
    @Override  
    public void printInfo(){  
        System.out.println("I am Robot");  
    }  
}
```

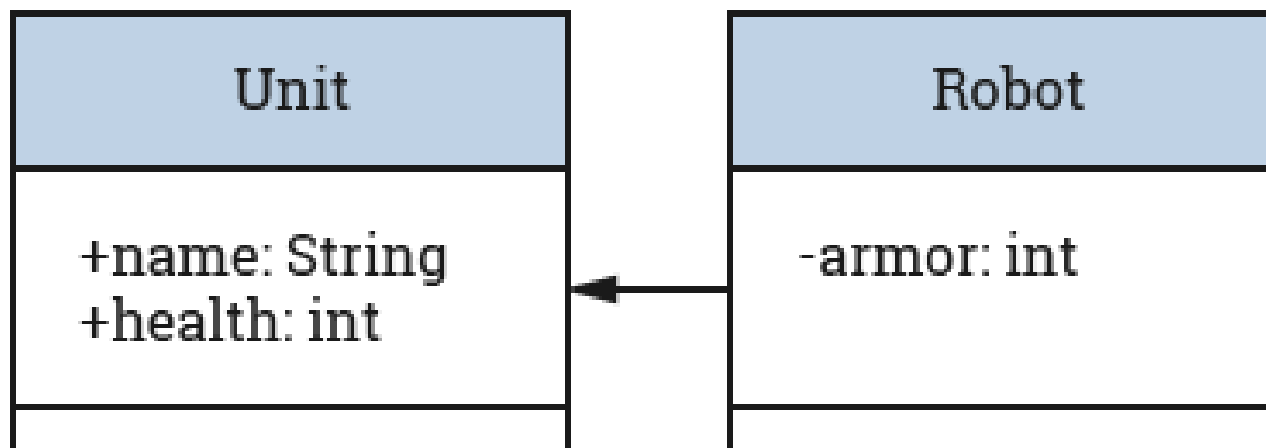


Графическое описание структуры классов в UML

○○○ ○○○

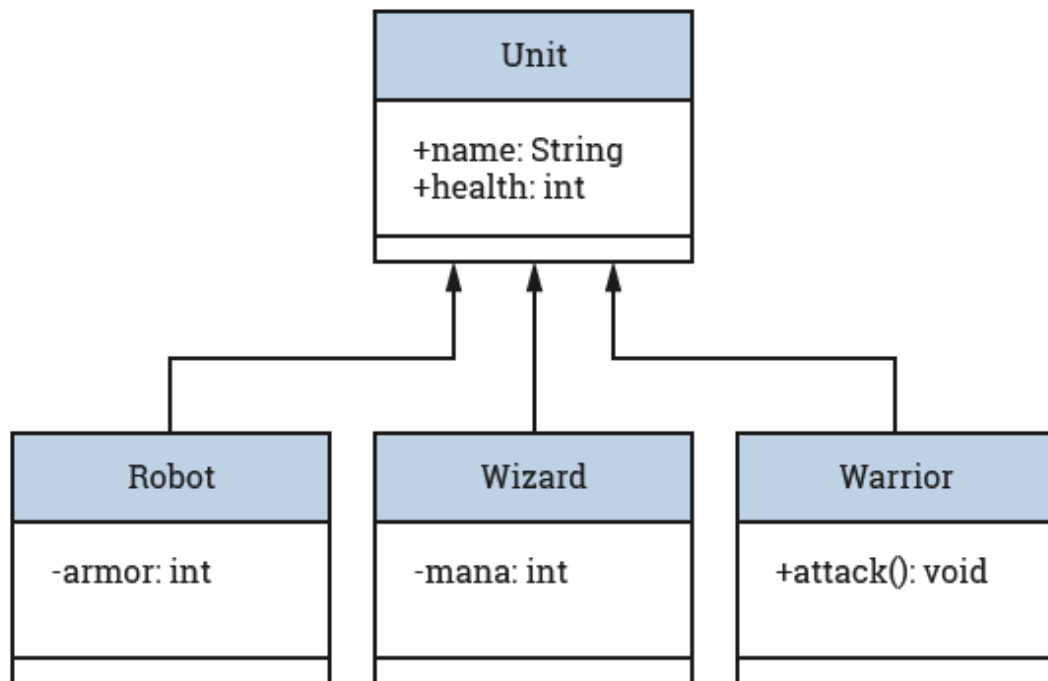


+ – public (публичный);
– private (приватный);
– protected (защищенный).



Графическое описание структуры классов в UML

○○○ ○○○



```
class Unit {  
    public String name;  
    public int health;  
}
```

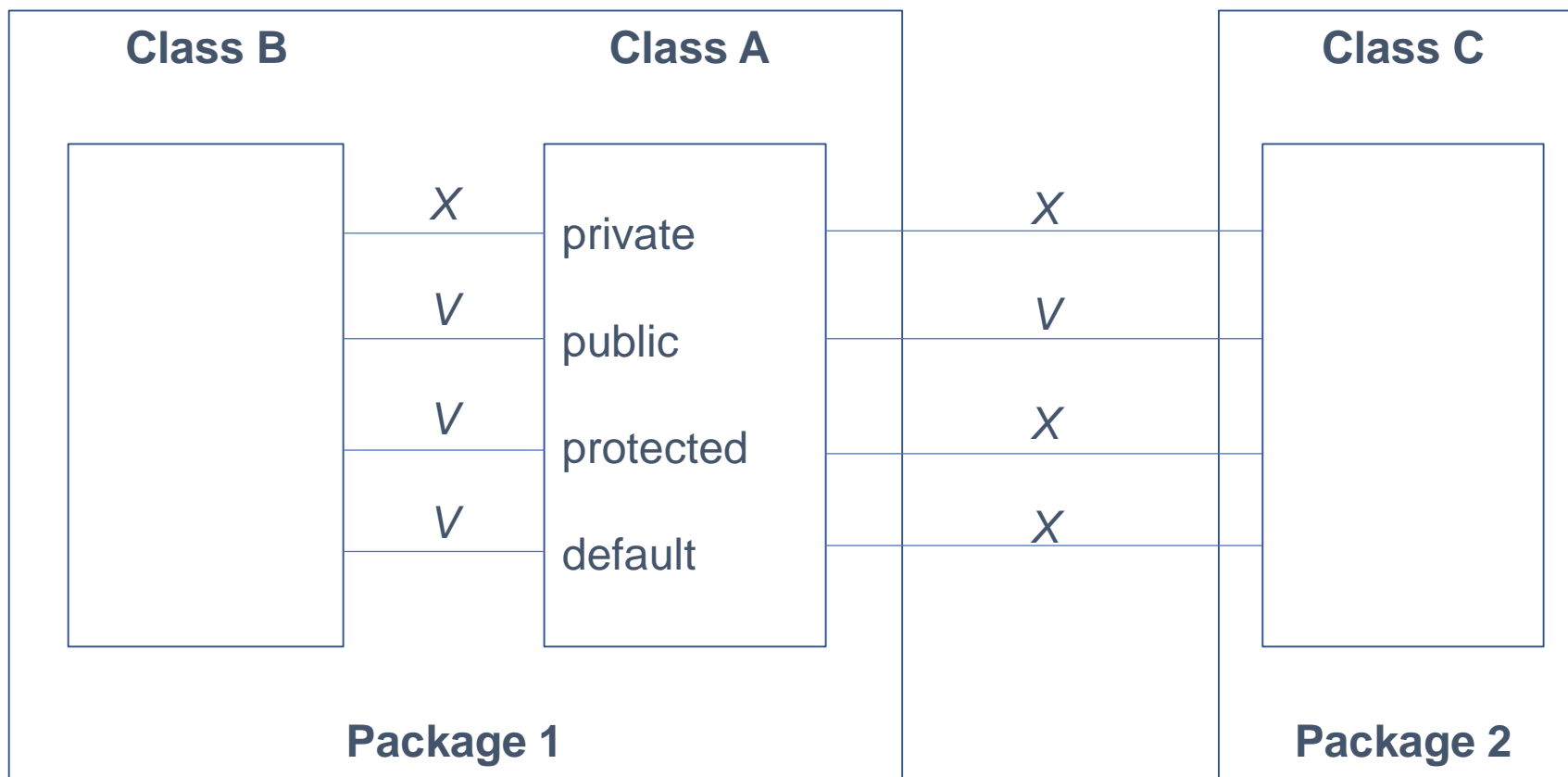
```
class Robot extends Unit {  
    private int armor;  
}
```

```
class Wizard extends Unit {  
    private int mana;  
}
```

```
class Warrior extends Unit {  
    public void attack() {}  
}
```

Ключевое слово protected

○○○ ○○○



Ключевое слово super

○○○ ○○○

```
class Robot extends Unit{  
    int armor;  
  
    public Robot(String name, int health, int armor) {  
        super(name, health);  
        this.armor = armor;  
    }  
}
```


```
class Robot extends Unit{  
    int armor;  
  
    public Robot(String name, int health, int armor) {  
        super(name, health);  
        this.armor = armor; }  
  
    public void printInfo(){  
        super.printInfo();  
        System.out.println("Armor:" + armor);  
    }  
}
```

Понятие полиморфизма

○○○ ○○○

```
Unit [] units = new Unit[5];
for(int i=0;i<units.length;i++){
    switch(new Random().nextInt(3)){
        case 0: units[i]=new Robot("Robot", 100, 100); break;
        case 1: units[i]=new Wizard("Wizard", 99, 80); break;
        case 2: units[i]=new Terminator("Terminator", 150, 150, "AK-47");break;
    }
}

for(Unit u: units) {
    u.printlnInfo();
    System.out.println();
}
```



Name:Wizard Health:99 Mana:80 Name:Terminator
Health:150 Armor:150 Gun:AK-47 Name:Wizard Health:99
Mana:80 Name:Wizard Health:99 Mana:80 Name:Robot
Health:100 Armor:100

Абстрактные классы

○○○ ○○○

```
abstract class Unit{  
    public String name;  
    public int health;  
  
    public abstract void printInfo();  
}  
  
class Robot extends Unit{  
    int armor;  
    @Override  
    public void printInfo() {  
        System.out.println("Name:" + name);  
        System.out.println("Health:" + health);  
        System.out.println("Armor:" + armor);  
    }  
}
```

Ключевое слово final

○○○ ○○○

Java Final Keyword

- ⇒ Stop Value Change
- ⇒ Stop Method Overriding
- ⇒ Stop Inheritance

javatpoint.com

```
public class MyProgram {  
    public final int MAGIC_CONST = 100;  
}
```

```
public class Example{  
    public final int helloWorld() {  
        System.out.println("Hello world!!!");  
    }  
}
```

```
public final class Example{  
}
```

Понятие интерфейса

○○○ ○○○

```
interface Playable{  
    void move();  
    void attack();  
}
```

```
class Robot implements Playable{  
    public void move() {  
        System.out.println("ride");  
    }  
  
    public void attack() {  
        System.out.println("shoot");  
    }  
}
```

```
class Wizard implements Playable{  
    public void move() {  
        System.out.println("walk");  
    }  
  
    public void attack() {  
        System.out.println("magic shoot");  
    }  
}
```

Иерархия наследования и преобразование типов

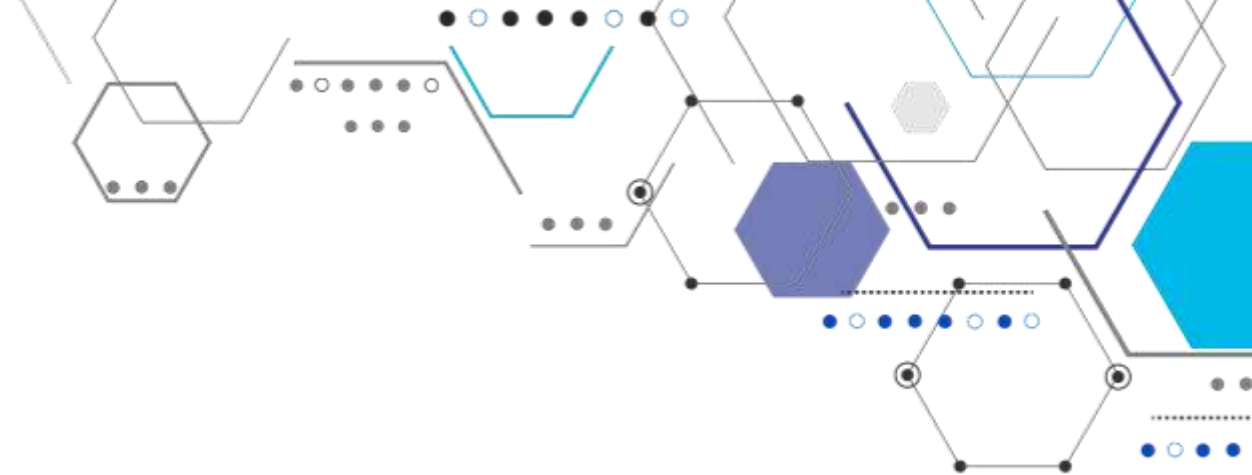
○○○ ○○○

```
Object object = new Robot("R2D2", 100, 100);  
Unit unit = new Wizard("Gendalf", 100, 1000);
```

//y Object нет метода printlnInfo, поэтому необходимо привести к Robot
`((Robot)object).printlnInfo();`

//y Unit есть метод printlnInfo
`unit.printlnInfo();`

//y Unit нет метода getMana, поэтому необходимо привести к Wizard
`int num = ((Wizard)unit).getMana();`



Спасибо за внимание!

