

IT ШКОЛА SAMSUNG

Интерфейсы

Модуль 2. Объектно-ориентированное программирование



Внутренние классы

Класс, который находится внутри другого класса, называется **вложенным**



Нестатический вложенный класс называется **внутренним**

Агрегация - это отношение, которое имеет место между несколькими классами в том случае, если один из классов представляет собой некоторую сущность, включающую в себя в качестве составных частей другие сущности

Внутренние классы

```
public class Computer {  
    class Processor {  
        //внутренний класс 1  
        private boolean isStart = false;  
        public void start() { isStart = true;}  
        public void shutdown() { isStart = false;}  
    }  
    class RAM {  
        //внутренний класс 2  
        private boolean isStart = false;  
        public void start() { isStart = true;}  
        public void shutdown() { isStart = false;}  
    }  
}
```

```
Processor i5 = new Processor(); //экземпляр процессора
```

```
RAM kingstone = new RAM(); //экземпляр оперативной памяти
```

```
Computer computer =  
    new Computer();  
  
computer.i5.start();  
computer.kingstone.start();
```



Анонимный класс

Внутренний класс, не имеющий имени

Использует только конструктор по умолчанию

```
new Computer() {  
    void superStart() { //НОВЫЙ МЕТОД  
        this.i5.start();  
        this.kingstone.start();  
    }  
};
```



Абстрактный класс

Группа методов, которые не могут быть реализованы в данном классе, но которые могут быть написаны в классах наследниках

- Все методы абстрактны
- Не возможности создать объект
- Можно создавать объекты классов-наследников

```
public abstract class A {  
    abstract public void draw();  
}
```

```
public class B extends A {  
    public void draw() {  
        тело метода ; }  
}  
A a = new B();
```

Реализация множественного наследования в Java:

Интерфейс

Описание типа, не имеющего объектов, но наследуемого классами.

Содержит поля и заголовки методов.

Все методы - абстрактные

```
public class В implements А {
```

Пользователь

...

Интерфейс

применяет

Activity как обработчик событий

```
public class MainActivity extends Activity implements OnClickListener{
    TextView tx;
    Button b;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        tx = (TextView)findViewById(R.id.tx);
        b = (Button)findViewById(R.id.b1);
        b.setOnClickListener(this);
    }
    @Override
    public void onClick(View arg0) {
        tx.setText("обработка");
    }
}
```

Задание:

В проекте SquareEqvation замените обработчик I1 на MainActivity

Создание интерфейса

```
public interface N { //здесь располагаются объявления полей и
методов
}
```

```
// создаем интерфейс
public interface MyI {
    int Get(int i);
    int Add(int value);
    int Size();
}
```

```
// применяем MyI
public class A implements MyI {
    @Override
    public int Get(int i) {
        return i; }
    @Override
    public int Add(int value) {
        return 0; }
    @Override
    public int Size() {
        return 0; }}}
```