Methods

Static Members

```
class Item {
  public static double tax = 0.2;
  public double getPrice(double inputPrice) {
    double margin = 0.05;
    return inputPrice * (1 + tax) * (1 + margin);
  }
}

public class MyClass {
  public static void main(String[] args) {
    Item item1 = new Item();
    System.out.println(item1.getPrice(100));
    Item item2 = new Item();
    item2.tax = 0.1;
    System.out.println(item2.getPrice(100));
    System.out.println(item1.getPrice(100));
 }
}
```

tax=0.2

tax=0.1

tax=0.1

tax belongs to a class

(not to an instance of the class)

tax=0.1

for all instances of the class!

```
126

115.5

115.5
```

```
// static members can be accessed directly
// (you don't need to create an instance of the object to access them)
class Dog {
  public static void barks() {
    System.out.println("Woof!");
  }
}

public class MyClass {
  public static void main(String[] args) {
    Dog.barks();
  }
}
```

```
// also possible
Dog dog = new Dog();
dog.barks();
```

```
// static methods can only call static fields directly
public class MyClass {
  String hi = "Good Afternoon!";
  public static void greet1() { System.out.println("Hello!"); }
  private static void greet2() { System.out.println(hi); }
  private void greet3() { System.out.println("Good Day!"); }
  public static void greetAll() {
    greet1();
    greet2();
    greet3();
  }
  public static void main(String[] args) {
    MyClass.greetAll();
  }
}
```

non-static

non-static

```
// fix #1 - make the fields static
public class MyClass {
  String static hi = "Good Afternoon!";
  public static void greet1() { System.out.println("Hello!"); }
  private static void greet2() { System.out.println(hi); }
  private static void greet3() { System.out.println("Good Day!"); }
  public static void greetAll() {
    greet1();
    greet2();
    greet3();
  }
  // main method
}
```

```
// fix #2 - create an instance for each call
public class MyClass {
  String hi = "Good Afternoon!";
  public static void greet1() { System.out.println("Hello!"); }
  private static void greet2() { System.out.println(new MyClass().hi); }
  private void greet3() { System.out.println("Good Day!"); }
  public static void greetAll() {
    greet1();
    greet2();
    new MyClass().greet3();
  }
  // main method
}
```

```
// fix #3 - create one one instance and use it every time
public class MyClass {
  String hi = "Good Afternoon!";
  static MyClass myClass = new MyClass();
  public static void greet1() { System.out.println("Hello!"); }
  private static void greet2() { System.out.println(myClass.hi); }
  private void greet3() { System.out.println("Good Day!"); }
  public static void greetAll() {
    greet1();
    greet2();
    myClass.greet3();
  }
  // main method
 }
```

```
// constants are usually marked static final and written in SNAKE_CASE


public class Item() {
  public static final double VALUE_ADDED_TAX = 0.25;
  public double calculatePrice (double price) {
    return price + price * VALUE_ADDED_TAX;
  }
}
```

```
// static final fields must be initialized before use
// this could be done in a static block

public class Item() {
    public static final double VALUE_ADDED_TAX;
    static {
      VALUE_ADDED_TAX = 0.25;
    }
    public double calculatePrice (double price) {
      return price + price * VALUE_ADDED_TAX;
    }
}
```

static block

is evaluated only once

```
// static blocks are useful when you need to calculate values


public class MetalBlock() {
  public static final double MASS_IN_KILOS;
  public static final double VOLUME_IN_CUBIC_METERS;
  public static final double DENSITY_IN_KILOS_PER_CUBIC_METER;
  static {
    MASS_IN_KILOS = 100;
    VOLUME_IN_CUBIC_METERS = 0.01;
  }
  static {
    DENSITY_IN_KILOS_PER_CUBIC_METER = MASS_IN_KILOS / VOLUME_IN_CUBIC_METERS;
  }
}
```

1

2

```
// static imports are used to import static members of classes

System.out.println(Math.pow(2, 5));

// pow() is the static method, so we can use static import
import static java.lang.Math.pow;
System.out.println(pow(2, 5));

// be careful about the order
// => import must be at the beginning

static import java.lang.Math.pow;
   => DOES NOT COMPILE
```