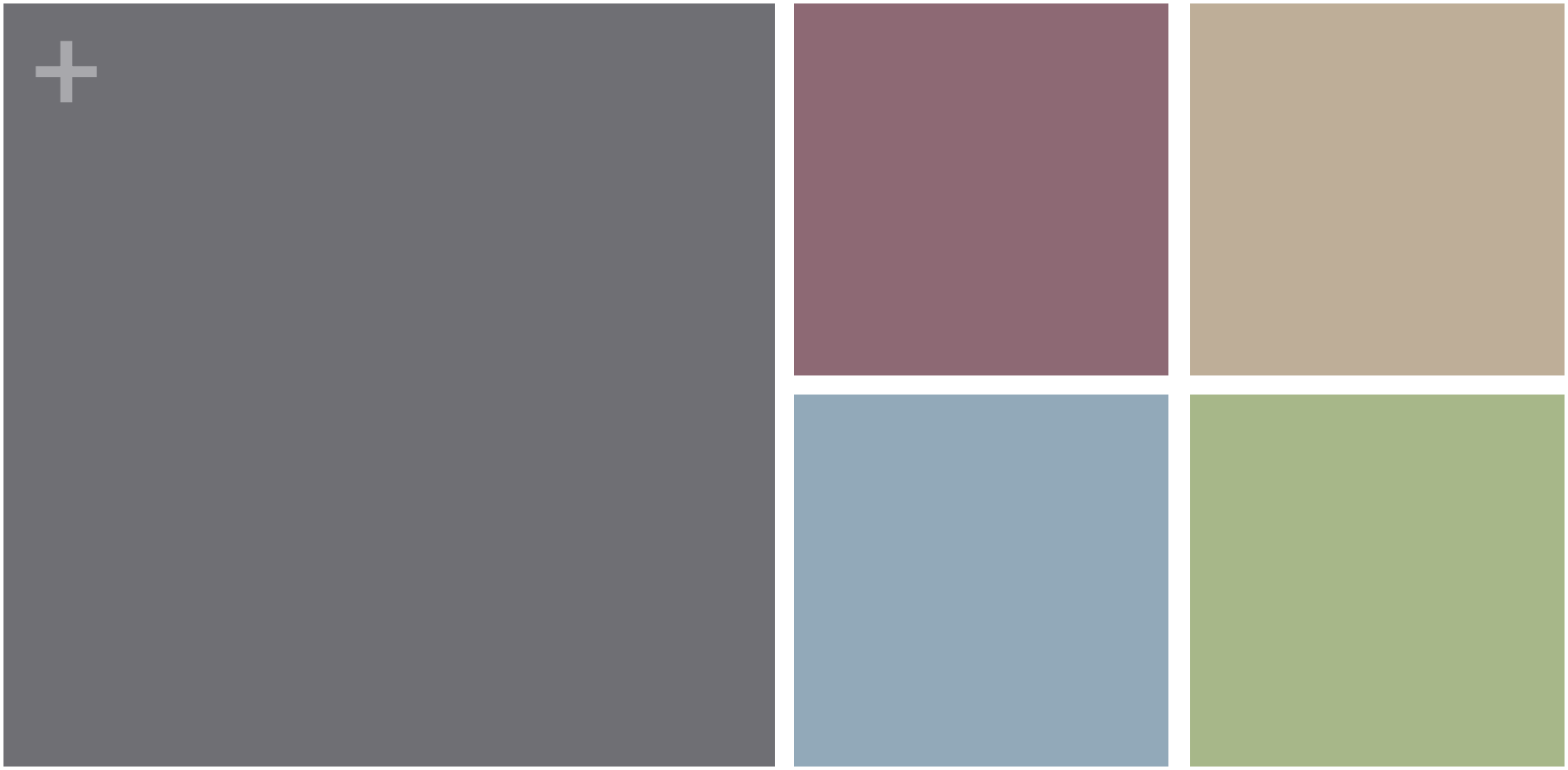

Computer programming (1)

Chapter 4

Static

Agenda

- **Static**
  - Static Instance Variable
  - Static Method
  - Static Import
  - Math Class

Static Variables

- A ***static variable*** is a variable that belongs to the class as a whole, and not just to one object
  - There is only one copy of a static variable per class, unlike instance variables where each object has its own copy
- All objects of the class can read and change a static variable
- Although a static method cannot access an instance variable, a static method can access a static variable
- A static variable is declared like an instance variable, with the addition of the modifier static

  ```
  private static int myStaticVariable;
  ```

- Static variables are also known as Class Variables.
- These variables can be accessed in any other class using class name.

Static Variables

- Static variables can be declared and initialized at the same time

  ```
  private static int myStaticVariable = 0;
  ```

- If not explicitly initialized, a static variable will be automatically initialized to a default value
  - **boolean** static variables are initialized to **false**
  - Other primitive types static variables are initialized to the zero of their type
  - Class type static variables are initialized to **null**
- It is always preferable to explicitly initialize static variables rather than rely on the default initialization

Static Variables

- A static variable should always be defined private, unless it is also a defined constant
  - The value of a static defined constant cannot be altered, therefore it is safe to make it **public**
  - In addition to **static**, the declaration for a static defined constant must include the modifier **final**, which indicates that its value cannot be changed

```
public static final int BIRTH_YEAR = 1954;
```

- When referring to such a defined constant outside its class, use the name of its class in place of a calling object

```
int year = MyClass.BIRTH_YEAR;
```

Example : Static variables

```
public class Student {

    String name;
    static int numOfStud=77;

    public int getNumOfStud() {
        return numOfStud;
    }

}
```

```
public class TestStudent {

    public static void main(String[] args) {

        Example obj = new Example();
        Example obj2 = new Example();

        Example.numOfStud = 66;
        //Example.Var2;   error

        System.out.println(obj.getVar1());
        System.out.print(obj2.getVar1());
    }
}
```

Static Methods

- A *static method* is one that can be used without a calling object
- A static method still belongs to a class, and its definition is given inside the class definition
- When a static method is defined, the keyword **static** is placed in the method header

```
public static returnedType myMethod(parameters)
{ . . . }
```

- Static methods are invoked using the class name in place of a calling object

```
returnedValue = MyClass.myMethod(arguments);
```

- Static Methods can access class variables (static variables) without using object of the class.
- It can access non-static methods and non-static variables by using objects.

Pitfall: Invoking a Nonstatic Method Within a Static Method

- A static method cannot refer to an instance variable of the class, and it cannot invoke a nonstatic method of the class
  - A static method has no **this**, so it cannot use an instance variable or method that has an implicit or explicit **this** for a calling object
  - A static method can invoke another static method, however

Example : Static method

```
public class Calculator {
    static int x=6;
    static int y=4;

    public static int add() {
        return x+y;
    }

    public int sub() {
        return x-y;
    }

}
```

```
public class TestCalculator {

    public static void main(String[] args) {
        Calculator c=new Calculator();

        System.out.println(Calculator.add());
        System.out.println(c.sub());
        System.out.println(c.add());
//System.out.println(Calculator.sub()); error

    }
}
```

Example : Static method

```
public class Student {

    String name;
    static int numOfStud=77;

    public int getNumOfStud() {
        return numOfStud;
    }

    public static int getName() {
        return name;
    }/* non-static variable name
cannot be referenced from a
static context */
}
```

```
public class TestStudent {

    public static void main(String[] args) {

        Example obj = new Example();
        Example obj2 = new Example();

        Example.numOfStud = 66;
        //Example.Var2;   error

        System.out.println(obj.getVar1());
        System.out.print(obj2.getVar1());
    }
}
```

```
/**
Class with static methods for circles and spheres.
*/
public class RoundStuff
{
    public static final double PI = 3.14159;

    /**
     Return the area of a circle of the given radius.
    */
    public static double area(double radius)
    {
        return (PI*radius*radius);
    }

    /**
     Return the volume of a sphere of the given radius.
    */
    public static double volume(double radius)
    {
        return ((4.0/3.0)*PI*radius*radius*radius);
    }
}
```

Static Import

Static imports are used to save your time and typing. If you hate to type same thing again and again then you may find such imports interesting

Example 1: Without Static Imports

```
class Demo1{
   public static void main(String args[])
   {
      double var1= Math.sqrt(5.0);
      System.out.println("Square of 5 is:"+ var1);
}
}
```

Output:

Square of 5 is:2.23606797749979

Example 2: Using Static Imports

```
import static java.lang.Math.*;
class Demo2{
   public static void main(String args[])
   {
      //instead of Math.sqrt need to use only sqrt
      double var1= sqrt(5.0);
      System.out.println("Square of 5 is:"+var1);
}
}
```

Output:

Square of 5 is:2.23606797749979

Tip: You Can Put a `main` in any Class

- Although the main method is often by itself in a class separate from the other classes of a program, it can also be contained within a regular class definition
  - In this way the class in which it is contained can be used to create objects in other classes, or it can be run as a program
  - A main method so included in a regular class definition is especially useful when it contains diagnostic code for the class

Example

```
public class Student {
    String name;
    static int numOfStud=77;

    public int getName() {
        return name;
    }

    public static int getNumOfStud() {
        return numOfStud;
    }

    public static void main(String[] args) {
            System.out.println(getNumOfStud());
    }
}
```

The Math Class

- The **Math** class provides a number of standard mathematical methods
  - It is found in the **java.lang** package, so it does not require an **import** statement
  - All of its methods and data are static, therefore they are invoked with the class name **Math** instead of a calling object
  - The **Math** class has two predefined constants, **E** (e, the base of the natural logarithm system) and **PI** (π, 3.1415 . . .)

    area = Math.PI * radius * radius;

Some Methods in the Class `Math` (Part 1 of 5)

Display 5.6 Some Methods in the Class `Math`

The `Math` class is in the `java.lang` package, so it requires no `import` statement.

```
public static double pow(double base, double exponent)
```

Returns `base` to the power `exponent`.

EXAMPLE

`Math.pow(2.0,3.0)` returns `8.0`.

(continued)

Some Methods in the Class `Math` (Part 2 of 5)

Display 5.6 Some Methods in the Class `Math`

```
public static double abs(double argument)
public static float abs(float argument)
public static long abs(long argument)
public static int abs(int argument)
```

Returns the absolute value of the `argument`. (The method name `abs` is overloaded to produce four similar methods.)

EXAMPLE

`Math.abs(-6)` and `Math.abs(6)` both return 6. `Math.abs(-5.5)` and `Math.abs(5.5)` both return `5.5`.

```
public static double min(double n1, double n2)
public static float min(float n1, float n2)
public static long min(long n1, long n2)
public static int min(int n1, int n2)
```

Returns the minimum of the arguments `n1` and `n2`. (The method name `min` is overloaded to produce four similar methods.)

EXAMPLE

`Math.min(3, 2)` returns 2.

(continued)

Some Methods in the Class `Math` (Part 3 of 5)

Display 5.6 Some Methods in the Class `Math`

```
public static double max(double n1, double n2)
public static float max(float n1, float n2)
public static long max(long n1, long n2)
public static int max(int n1, int n2)
```

Returns the maximum of the arguments n1 and n2. (The method name max is overloaded to produce four similar methods.)

EXAMPLE

`Math.max(3, 2)` returns 3.

```
public static long round(double argument)
public static int round(float argument)
```

Rounds its `argument`.

EXAMPLE

`Math.round(3.2)` returns 3; `Math.round(3.6)` returns 4.

(continued)

Some Methods in the Class `Math` (Part 4 of 5)

Display 5.6 Some Methods in the Class `Math`

```
public static double ceil(double argument)
```

Returns the smallest whole number greater than or equal to the `argument`.

EXAMPLE

`Math.ceil(3.2)` and `Math.ceil(3.9)` both return `4.0`.

(continued)

Some Methods in the Class `Math` (Part 5 of 5)

Display 5.6 Some Methods in the Class `Math`

```
public static double floor(double argument)
```

Returns the largest whole number less than or equal to the `argument`.

EXAMPLE

`Math.floor(3.2)` and `Math.floor(3.9)` both return `3.0`.

```
public static double sqrt(double argument)
```

Returns the square root of its `argument`.

EXAMPLE

`Math.sqrt(4)` returns `2.0`.

+

Thanks!