OL Academy

ITCS107/114

Computer Programming II

Chapter (6)

Java Programming Review
Classes & Methods

Before we begin
(Review of classes & methods)

Syntax of declaration a class

```
public class className
{
 // Data fields declarations (member variables)

 // Constructors

 // Methods
}
```

class body

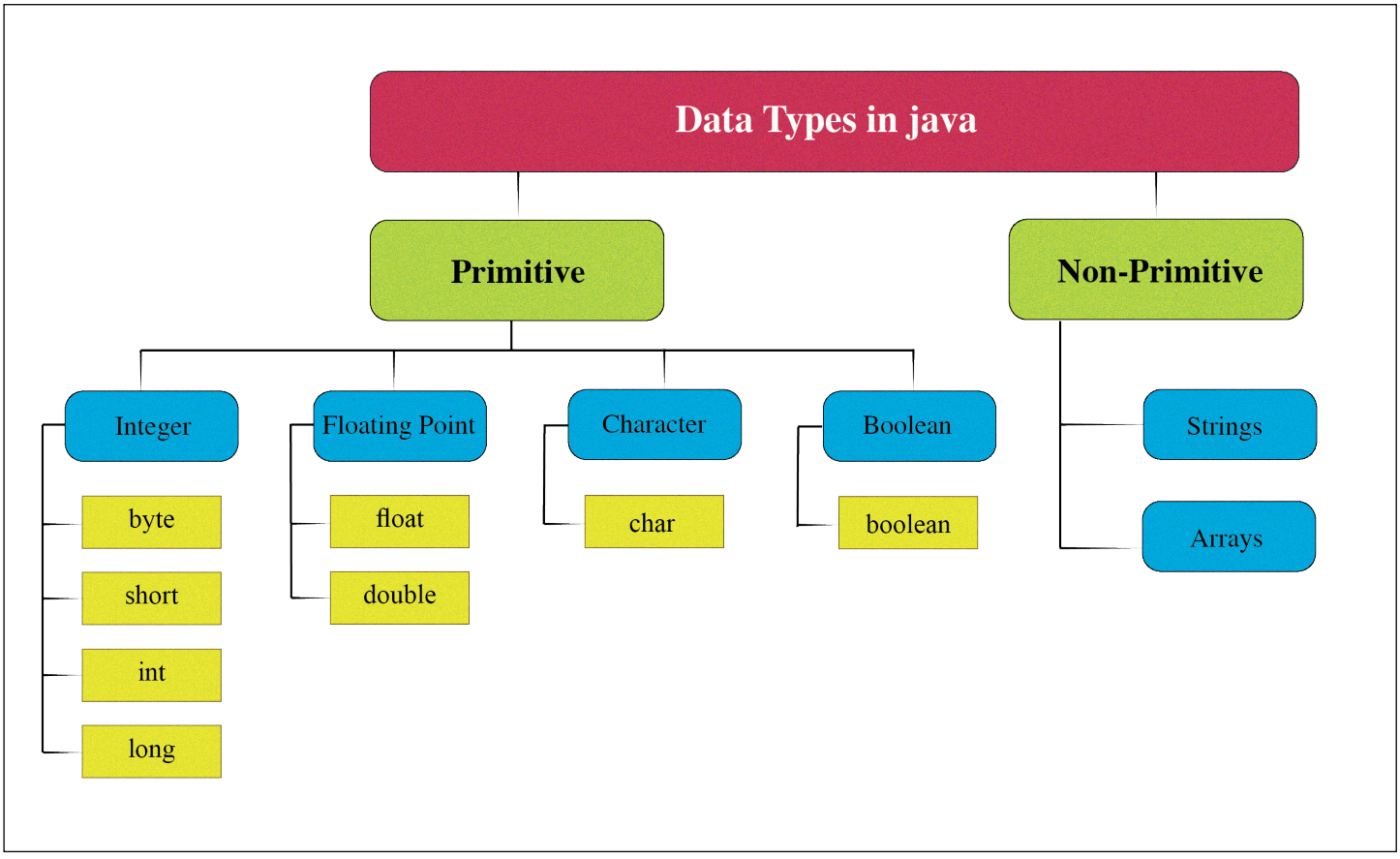
Note that:

- A class is declared by use of **class** keyword.
- The class body is enclosed between curly braces { }.
- The class can have only two access modifiers:
  - **public:** class is visible to all classes everywhere.
  - **default (no modifier):** it is visible only within its own package.
- The data or variables defined within a class are called **instance variables**.
- The methods and variables defined within a class are called member of the class.

Data types in Java:

Data types are divided into two groups:

- **Primitive data types:**
  There are 8 primitive data types such as byte, short, int, long, float, double, char and boolean.
- **Non-primitive data types (object data types):**
  such as Strings and Arrays.



All of Java primitive types with amount of computer memory they use (size):

Type Name	Kind of value	Size	Examples
byte	Integer	1 byte	`byte x = 3;`
short	Integer	2 bytes	`short x = 3;`
int	Integer	4 bytes	`int x = 3;`
long	Integer	8 bytes	`long x =3;`
float	Floating Point	4 bytes	`float x =3.5;`
double	Floating Point	8 bytes	`double x = 3.5;`
char	Single Character	2 bytes	`char x ='O';`
boolean		1 bit	`boolean x= false;`

Remember that: some of Escape Characters using with Strings in java	
\"	double quote
\'	Single quote
\\	Backslash
\n	New line. Go to the beginning of the next line.
\t	Tab. Add whitespace

Declaration of instance variables:

- Instance variables are declared in a class, but outside a method, constructor or any block.
- An instance variable can be declared with these access modifiers:

  - default (no modifier): accessible only by classes in the same package.

  - public: visible to any class, whether these classes are in the same packages or in another package.

  - private: The member can only be accessed in its own class.

  - protected: The member can be accessed within its only own package.

Examples of declaration a variable

```
long cprNum; //without access modifier

public long cprNum; //with public access modifier

private long cprNum; //with private access modifier

protected long cprNum; //with protected access modifier
```

Static variables (Class variables)

- Static variables are declared with the **static** keyword in a class (outside a method).
- Static variables are shared by all objects of a class.
- Static variables that are not constants should be private.
- Can be accessed by calling with the class name (doesn't need any object).

```
className.variableName;
```

- Variables declared **static final** are considered constant value (cannot be change).

```
public static final int WEEKDAYS = 7;
```

- We can have a static variables that can change in value, they are declared like instance variables but with the keyword **static**.

```
private static int numberOfVacations;
```

- Both static variables and instance variables are sometimes called **fields** or **data member**.

Remember: **Java has three kinds of variables**

- Instance variables
- Static variables
- Local variables

Advantages of static variable:
It makes your program memory efficient (It saves memory)

Declaration of methods:

- A method is a block of statements that has a name and can be executed by calling (invoking) it.
- Every program must have at least one method, and every program must have a methed named **main**, which is the method first invoked when the program is run.
- All methods ***-including the main-*** must begin with a method declaration.

Syntax of declaration a method

```
public ReturnType methodName(parameter(s))
{

 // body of the method

}
```

- Variables in a method are called **Local variables**.
- Local variables having the same name and declared in different methods are different variables.

Static methods:

static methods are the methods in java that can be called without creating an object of class.

```
ClassName.methodName( );
```

Instance method (non static) & static method

	instance (variables & methods)	static (variables & methods)
instance method	can access directly	can access directly
static method	can't access directly (must use reference to object)	can access directly

(1)
Classes and Objects

Syntax:

```
public class ClassName
{
    // attributes and methods
}


public class Test
{
     public static void main (String[]args)
     {
             // define objects
     }
}
```

Example (1):

(A) write a class called **Person** have the following data members (Private): name(string), cpr(1ong)

and the following methods:

- Set and get methods for name and cpr.
- print method.

```
Class Person
```

```
public class Person
{
    private String name;
    private long cpr;
```

```
    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setCpr(long cpr) {
        this.cpr = cpr;
    }

    public long getCpr() {
        return cpr;
    }

    public void print ( ) {
        System.out.println ("Name : "  + name + "\nCPR : " +
cpr);
        }

} //end of class Person
```

(B) Write a class TestPerson having only main method. In the main method declare two objects p1 and p2 of type person. create an object of type Person having some suitable values of attributes and assign it to variable p1 .

main Class

```
import java.util.Scanner;
public class TestPerson {
    public static void main(String[] args) {
        Person p1 = new Person();
        p1.print();       >> null   0

        Person p2 = new Person();
        p2.setName("Marwa");
        p2.setCpr(123456789);

        p2.print();  >> Marwa   123456789

        Person p3 = new Person ();
        Scanner kbd = new Scanner (System.in);
        System.out.println("Enter name:");
        String name;
        name = kbd.nextLine();  >> Ahmed

        p3.setName(name);

        System.out.println("Enter cpr:");
        long cpr;
        cpr = kbd.nextLong(); >> 112233445
        p3.setCpr(cpr);

        p3.print(); >> Ahmed    112233445


} //end of class Test
```

Examples : What is the output of the below code segment?

A)

```
public class Student {
    private String name;
    private double gpa;
    private static int noOfStudent = 0;

    public Student (){
        name = " ";
        gpa =0;
        noOfStudent++;}

    public Student (String n, double g){
        name = n;
        gpa =g;
        noOfStudent++;}

    public static int getNoStudent() {
        return noOfStudent;}


    public void noOfStudent(){
        System.out.println(noOfStudent);
    }

    public void print()
    {
        System.out.println("Name: " + name);
        System.out.println("GPA: " + gpa);
        System.out.println("Number of Students: " + noOfStu-
dent);
    }
}
```

```
public class TestStudent {
    public static void main(String[] args) {
        Student st1 = new Student();
        st1.noOfStudent();
        Student.getNoStudent();

        Student st2 = new Student("Ali",3.2);
        st2.noOfStudent();
        st1.noOfStudent();

        Student st3 = new Student("Ebrahim",3.8);
        st3.noOfStudent();
        st1.print();
        st2.print();
        st3.print();

    }
}
```

Output

```
1
2
2
3
Name:
GPA: 0.0
Number of Students: 3
Name: Ali
GPA: 3.2
Number of Students: 3
Name: Ebrahim
GPA: 3.8
Number of Students: 3
```

B-

```
public class Variable {
    private static int count=0;
    private static int nbVariables=0;
    private int value;

    public Variable (int newValue) {
        value = newValue;
        nbVariables++;
    }

    public void increment () {
        count++;
    }

    public int getVariable() {
        return nbVariables;
    }

    public int getValue() {
        return value;
    }

    public int getCount(){
        return count;
    }
}
```

```
public class VariableOutput
{
    public static void main(String[] args) {
        Variable obj1 = new Variable(5);

        Variable obj2 = new Variable(5);

        int n=20;
        Variable obj3 = new Variable(n);

        obj1.increment();
        obj2.increment();
        System.out.println(obj2.getVariable());
        System.out.println(obj3.getValue());

        obj3.increment();
        System.out.println(obj3.getCount());

    }
}
```

Output

```
3
20
3
```

Question (1): Consider a Class named **App** with the following private data members:
name: to represent the app's name. The default value is "NewApp".
rate: to represent the app's rate, in a range from 0.0 to 5.0. The default value is 0.0.
downloads: to represent the app's number of downloads, as a positive number. The default value is 5.

In addition, the class has the following methods:
A- default constructor to initialize the data members.
B- set method named setApp to set all app information. Please provide an error message for invalid values.
C- get method for each data member.
D- method named isTopApp that returns true if the app has a rate of 5.0 and more than 100 downloads. Otherwise the method should return false.

Write the App Java Class.

```
impport java.util.Scanner;
public class App
{
    private String name;
    private double rate; // The default value is 0.0
    private int downloads; //positive number. The default
value is 5.
```

A- default constructor :

```
public App()
{
        name = "NewApp";
        rate = 0;
        downloads = 5;
}
```

B- set method

```
public void setApp(String n,double r,int d)
{
    name = n;

    if(r >= 0 && r <= 5)
        rate = r;
    else
        System.out.println("Invalid Rate!");

    if(d < 0)
        System.out.println("Invalid Downloads!");
    else
        downloads = d;
}
```

C- get method for each data member.

```
public String getName() {
   return name;
}

public double getRate() {
   return rate;
}

public int getDownloads()
{
   return downloads;
 }
```

D-isTopApp method

```
public boolean isTopApp()
{
   return (rate == 5.0 && downloads > 100);
}

} // end of the class App
```

Question (2): Create a class named fruit with two private data members:

- **name**: representing a fruit name. Default value is “Fruit”.
- **price**: representing a fruit price as a non-negative floating point number. Default value is 1.0.

Additionally, the class should also include the following methods:

- A default constructor to initialize the data members.
- A set method with parameters to set name and price.
-Two get methods to return values of name and price.

```
public class Fruit {
    private String name;
    private double price;

    public Fruit( )
    {
        name = “Fruit”;
        price = 1.0;
    }

    public void setFruit(String n, double p)
    {
        name = n;
        if (p>0)
            price = p;
    }
    public String getName() {
        return name;
    }
    public double getPrice() {
        return price;
    }
}
```

Math Class

Java Math class provides several methods to work on math calculations like pow(), abs(), max(), min, ect..

(1) Math.pow() method: It used to return the value of first arguments raised to the power of the second argument, The return type of this method is double.

Syntax:

```
public static double pow (double a, double b)
```

Example:

```
System.out.Println(Math.pow(2,4));  //16.0

int x=2, y=2;
System.out.Println(Math.pow(y,x));  //4.0
```

(2) Math.abs() method: Returns the absolute value if a int,long,float, double value.

Syntax:

```
public static int abs (int a)
```

Example:

```
int x=55;
int y=-41;
System.out.Println(Math.abs(x));  //55
System.out.Println(Math.abs(y));  //41
System.out.Println(Math.abs(x/0));  //infinity
```

Static methods in class Math

Name	Description	Argument Type	Return Type	Example	Value Returned
pow	Power	double	double	Math.pow(2.0,3.0)	8.0
abs	Absolute value	int, long, float, or double	Same as the type of the argument	Math.abs(-7) Math.abs(7) Math.abs(-3.5)	7 7 3.5
max	Maximum	int, long, float, or double	Same as the type of the arguments	Math.max(5, 6) Math.max(5.5, 5.3)	6 5.5
min	Minimum	int, long, float, or double	Same as the type of the arguments	Math.min(5, 6) Math.min(5.5, 5.3)	5 5.3
random	Random number	none	double	Math.random()	Random number in the range ≥ 0 and < 1
round	Rounding	float or double	int or long, respectively	Math.round(6.2) Math.round(6.8)	6 7
ceil	Ceiling	double	double	Math.ceil(3.2) Math.ceil(3.9)	4.0 4.0
floor	Floor	double	double	Math.floor(3.2) Math.floor(3.9)	3.0 3.0
sqrt	Square root	double	double	Math.sqrt(4.0)	2.0

Overloading

Overloading occurs when two or more methons in one class have the same method name but different parameters.

```
public class Overload
{
    public static double getAverage (double a, double b)
    {
        return (a+b)/2;
    }

    public static double getAverage(double a,double b,double c)
    {
        return (a+b+c)/3;
    }

    public static void main(String[] args) {
        double avg1 = Overload.getAverage(40,50);
        System.out.println("Average1 = " + avg1);

        double avg2 = Overload.getAverage(2,3,5);
        System.out.println("Average2 = " + avg2);

    }
}
```

Output

```
Average1 = 45.0
Average2 = 3.3333333333333335
```

The method **getAverage** is overloading, we have two methods in the same class, but each method take different number of parameters or different types of parameters.