TP09

Java Class and Objects (part 2)

# Remark

1. Overloading methods – Are methods which has the same name but different arguments. Example:

**public** **class** Compare {

**int** **max**(**int** a, **int** b) {

**if** (a > b)

**return** a;

**else**

**return** b;

}

String **max**(String a, String b) {

**if** (a.compareTo(b) > 0)

**return** a;

**else**

**return** b;

}

}

Method max() is called overloading method, because there are 2 methods with the same name max.

1. Class Variable – a variable is common to all instance of this class. It means that all object instance of this class share this variable only one address memory. In Java, we use static keyword place in front of variable that we want it to become class variable.

Example:

**public** **class** UserTest {

**public** **static** **void** main(String[] args) {

System.***out***.println("user\_count = "+User.*user\_count*);

User u1 = **new** User("Dara");

System.***out***.println("user\_count = "+User.*user\_count*);

User u2 = **new** User("Sotha");

System.***out***.println("user\_count = "+User.*user\_count*);

User u3 = **new** User("Sothea");

System.***out***.println("user\_count = "+User.*user\_count*);

User u5 = **new** User("Nida");

System.***out***.println("user\_count = "+User.*user\_count*);

}

}

**public** **class** User {

**static** **int** *user\_count* = 0;

String username;

**public** User(String username){

**this**.username = username;

*user\_count*++;

}

}

Every time, we create a User object, the user\_count variable is increased by 1. The variable user\_count is created and initialized only one time. Output of program above:

user\_count = 0  
user\_count = 1  
user\_count = 2  
user\_count = 3  
user\_count = 4

1. Class Method – method that can be called without instantiating object of that class. In Java, we use static keyword in method declaration. Example: Math.abs(int), Math.sqrt(double)
2. Constants – variable that is not changeable at runtime. Example: Math.PI
3. Object Destruction – a special method named finalize() will be called automatically when an object is removed from memory.

# TP07.1. Rectangle

We have a class Rectangle as below:

**public** **class** Rectangle {

**int** width;

**int** height;

**public** Rectangle(**int** width, **int** height) {

**this**.width = width;

**this**.height = height;

}

**public** **int** calculatePerimeter(){

**return** (width + height) \* 2;

}

**public** **int** calculateSurface(){

**return** width \* height;

}

}

Create a class RectangleTest that is a Java application that test:

1. Create an instance of class Rectangle (call its constructor)
2. Display perimeter of it (the new created rectangle object)
3. Display surface of it (the new created rectangle object)

# TP07.2. SMS Encrypt

Create a Java class represents SMS. SMS class contains:

* Attributes: Subject, From phone number, Receiver number, Type (Text, MMS), content, and status (new, read)
* Constructor (s): suggests 3 constructors

Then create a class named SMSList, that represents list of SMS and manage SMS in and out:

* Attributes: ArrayList<SMS>, and static field max\_characters\_per\_sms
* Operations/Methods: suggests 3 overloaded methods

Then write a program that will display a menu:

1. Send new SMS with Encrypted content using password method
2. View SMS detail
3. List SMSes
4. Remove SMSes by index
5. Quit

# TP07.3. SMS Decryption

Using previous exercise as references and reuse classes of it. Create new project for SMS Decryption part. This part will be used by another user.

Then, write a program in Java will show a menu:

1. List all SMSes
2. View SMS Detail (decrypt content using password)
3. View readable SMSes (all SMS that can be decrypted using given password)
4. Remove SMSes by index
5. Quit