TP05

Java Loops

Remark

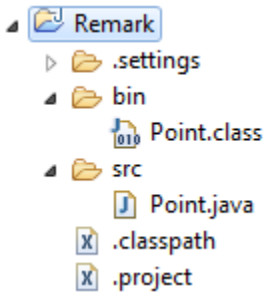
1. Class – A class can be defined as a template/blueprint that describes the behavior/state that the object of its type support.
2. Object – Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behaviors – wagging the tail, barking, eating. An object is an instance of a class.

Example class Point needs to have:

```
public class Point {
	int x;
	int y;
}
```

This class must be stored in file named: “Point.java”
After compiled successfully (without error), it has generated new file named: “Point.class”

In Eclipse, source files (*.java) are stored separate folder from compiled files (*.class). As in screenshot below:



The folder **bin** is used to store ***.class** files, and the folder **src** is used to store ***.java** files.

The class Point created previously has 2 attributes x and y and has no method.

This class can’t be run because it has no method main. All Java applications must have method main!!!

Without method main, it is a class, it is not an error even if it contains no method main. Java provide ability for other class (es) to access and instantiate it. Like in application below:

```java
public class ManagePoints {

	public static void main(String[] args) {
		Point p1 = new Point(); // object name p1 instantiated (new)
					// using default constructor
		p1.x = 2; // set attribute x of object name p1 to 2
		p1.y = 5; // set attribute y of object name p1 to 5

		Point p2 = new Point(); // object name p2 instantiated (new)
					// using default constructor
		p2.x = 8; // set attribute x of object name p2 to 8
		p2.y = 3; // set attribute y of object name p2 to 3

		// create variable distance to distance value. Now, initialize it 0
		double distance = 0;
		distance = Math.sqrt(
				Math.pow(p2.x - p1.x, 2) + Math.pow(p2.y - p1.y, 2)
			);

		System.out.printf("Distance between p1(%d,%d) and p2(%d,%d) is: %f",
				p1.x, p1.y, p2.x, p2.y, distance);
	}

}
```

The usage of class Point here is similar to C programming struct:

```c
struct Point {
   x:int;
   y:int;
};
typedef struct Point Point;
```

*** Constructors

Every class has a constructor. If we do not **explicitly** write a **constructor** for a class, the Java compiler builds a **default constructor** for that class.

Each time a new object is created, at least one constructor will be invoked. The main rule of constructors is that they should have the same name as the class. A class can have more than one constructor.

The class Point above has no explicit constructor, so Java compile builds a default constructor for it. How it look like? as you notice in method main of class ManagePoints, the expression:

```java
Point p1 = new Point();
```

The default constructor is a constructor without parameter. So, it looks like below:

```
public class Point {
	int x;
	int y;

	public Point() {

	}

}
```

Explicit constructors are constructors with parameter (s). So, it could be any form as below:

```
public class Point {
	int x;
	int y;

	public Point() {

	}

	public Point(int _x){
		x = _x;
		y = 0;
	}

	public Point(int _x, int _y){
		x = _x;
		y = _y;
	}
}
```

In the world of Objects, we might move the calculation of distance between 2 Points, from class ManagePoints into class Point so that calculation made more optimize. Like in example below:

```java
public class Point {
    int x;
    int y;

    public Point() {

    }

    public Point(int _x) {
       x = _x;
       y = 0;
    }

    public Point(int _x, int _y) {
       x = _x;
       y = _y;
    }
    public double distance(Point anotherPoint) {
          double distance = 0;
          distance = Math.sqrt(Math.pow(anotherPoint.x - x, 2)
                       + Math.pow(anotherPoint.y - y, 2));
          return distance;
    }
}
```

```java
public class ManagePoints {

   public static void main(String[] args) {
       Point p1 = new Point(); // object name p1 instantiated (new)
                                       // using default constructor
       p1.x = 2; // set attribute x of object name p1 to 2
       p1.y = 5; // set attribute y of object name p1 to 5

       Point p2 = new Point(); // object name p2 instantiated (new)
                                       // using default constructor
       p2.x = 8; // set attribute x of object name p2 to 8
       p2.y = 3; // set attribute y of object name p2 to 3
       // create variable distance to distance value. Now, initialize it 0

       double distance = p1.distance(p2);

       System.out.printf("Distance between p1(%d,%d) and p2(%d,%d) is: %f",
              p1.x, p1.y, p2.x, p2.y, distance);
   }

}
```

p1 itself is a Point, so we need only another Point p2. In p1 has attributes x and y.

TP05.1. Prime Number (review from TP04.1)

Write an application Java that let user input a number then list prime numbers from 2 to inputted number. Example:

```
Input number to list prime numbers from 2 to it: 29
2 3 5 7 11 13 17 19 23 29 is prime number.
```

TP05.2. Odd Numbers

Write a program in Java to display odd numbers (ex: 1, 3, 5, 7, etc.) located between 0 and 500.

TP05.3. Even Numbers

Write a program in Java to display even numbers (ex: 2, 4, 6, 8, etc.) located between A and 500. Where A is given by user (0<A<500).

TP05.4. Company Profits

Write a program in Java to calculate company profits for 12 months. Example:

```
Profit for month 1  : 25
Profit for month 2  : 2.5
Profit for month 3  : -63
Profit for month 4  : 0
Profit for month 5  : 10
Profit for month 6  : 32
Profit for month 7  : 26.5
Profit for month 8  : 28.2
Profit for month 9  : 29
Profit for month 10 : 31.7
Profit for month 11 : 29.5
Profit for month 12 : 26

Total profits for 12 months   : 177.40
```

TP05.5. Palindrome

Write a program in Java to check whether a given string is a palindrome. Palindrome are words or phrases that the same when reading from left to right or from right to left. The check method is chosen by user. Example:

```
Please gives a word to check: NOON
Choose method (REV, LOOP): REV
NOON is a Palindrome
```

Example 2:

```
Please gives a word to check: SOKKANG
Choose method (REV, LOOP): LOOP
SOKKANG is not a Palindrome
```

TP05.6. String Mirroring

Write a method makePalindrome() that creates a reverse copy of given String, and then join them to make a Palindrome.

Then, write a program in Java to create a reverse copy of given String. Example:

```
Please enter a word: DICE
DICEECID
```

TP05.7. Escape Characters Replacement

Write a program in Java to read a sentence from keyboard. Then replaces all escape characters follow below table:

Escape Character	Replacement
\n	(new_line)
\t	(tab)
****	(slash)
//	(comment_line)
:)	(smile)

For Example:

```
Please enter a sentence: \n \t is used to represent new line \n\\ it is a // line
comment :)

(new_line) (tab) is used to represent new line (new_line)(slash) it is a
(comment_line) line comment (smile)
```

SECTION BELOW IS BEGINNING OF CHALLENGE EXERCISES (+10% ABOVE THE TOTAL SCORES)

TP05.CE.1. RangeUtil Class

A Range is a sequence of numbers from a started point to an ending point. For example, a range starts from 5 to 10 is 5, 6, 7, 8, 9, 10. In this example, the number is increased 1 by 1. We can have it increase by 2 for example, 5 7 9. The size of increase is called step. Create class RangeUtil that contains:

- Constructors that initialize its attributes: start, end, and step
- Method toString() that return a string represents the range of numbers (e.g. “5, 6, 7, 8, 9, 10”)

Modify the Odd and Even program (TP05.2 and TP05.3) to use RangeUtil class instead.

Remark: The range can also start from greater number to smaller one for example, 10, 9, 8, 7, 6, 5.

TP05.CE.2. MessageCoder class

Text message is useful for communication nowadays. However, some messages can contain special characters that maybe translated into something like special signs or smileys. Those special characters are not intended to be input by user but we can’t stop user from inputting characters that lead to confusion or mis display those characters on the other end. Encoder is used to convert inputted text message into nonspecial characters text message (called Encoded Message). Decoder is used to convert Encoded Message back to original message.

Create class MessageCoder that can encode and decode message(s). Then, modify TP05.7 to use this class and add possibility to decode message back to original message.