Exercise 1:

1. Il écrit la commande pour imprimer (print) :

System.***out***.println();

1. Il nous donne plusieurs options et quand on choisi la toString(), il écrit :

@Override

**public** String toString() {

// **TODO** Auto-generated method stub

**return** **super**.toString();

}

1. Il écrit la méthode pour main :

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

}

1. Il nous donne plusieurs options dont : constructor, equals function, toString() function, …

**public** First() {

// **TODO** Auto-generated constructor stub

}

@Override

**public** **boolean** equals(Object obj) {

// **TODO** Auto-generated method stub

**return** **super**.equals(obj);

}

@Override

**public** String toString() {

// **TODO** Auto-generated method stub

**return** **super**.toString();

}

1. We can rename the class and all the refrences that refer to this class in the file. Same for the field foo.

Exercise 2:

1. It works because by default we have a constructor for Point that does not take any arguments

2. We have a compile error. It doesn't work because we have private variables x and y and they are only available in the class Point

To fix it we need to change the visibility of the variables of Point (x and y) to public or protected or add a getter method

3. To protect the values or function from unwanted actions, meaning the caller cannot usually get inappropriate direct access to the field

4. Accessor is a getter method that it is usually used when we have a private field, and we want to the values of this field in another class. So, we use a get method that return the value of the field.

Yes, we need to do it here because we have private values (x and y) and we need to access them form another function in another class

5. We have a compile time error because we now have a constructor taking variables and the default constructor that we had before is not the default one.

6. The program will be confused between the values that we have in the class and the values that we are passing as parameters, so we need to use the **this** keyword to define the values we have in the class

7. Create a variable called sum in the class and increment its value in constructors of point

8. If we pass by the function two doubles as parameters, the compiler will call this method:

**public** Point(**double** x,**double** y) {

and if we pass by only one point as a parameter, the constructor will call this method:

**public** Point(Point p2) {

Exercise 3:

1. Output:

True

False

Because in the first one both point to the same place in memory but in the second one each object points to a differ place in memory even though they both have the same values for x and y

1. Output:

0

-1

First the code printed 0 because in memory we have p2 pointing to p1 and indexOf checks where the object is using the reference in memory, so it printed 0

Then the program printed -1 to say that p3 is not in the list but actually p3 is the same as p1 so it should have printed 0

IndexOf checks if the point exists in the list using the equals method

When we override the equals method we got that both points p2 and p3 are in the list

Exercise 4:

2. If we add more points of the max capacity we will have an run time error

We need to throw an exception so that when we reach maximum capacity we can’t add values anymore

5. If we add null, it will add null in the array.

6. In an arrayList we don’t have a limit to the size of the list. So, pointCapacity is unlimited, nbPoints will be equal to the size of the list and the method contains will not change because we are using the for each loop.

Exercise 5:

1. we can increment the values of x and y by adding the value of dx to x and the value of dy to y

5. we are using the point p for both circles, so when we translate a circle we are changing the value of p, so both circles translate too. To avoid it we need to create a new point j with the new values of x and y and then equate p to j.

9. This method should be declared in the TestPoint class and not in the Circle class.

Exercise 6:

1. No

4. It will print the address of ring in memory and not the actual data that we want, we need to add a toString method.