Chap10Yourname.java

objects

```
/*
 * Chap 10 Objects
 * Java is an "object-oriented" language, which means that is uses objects
 * to represent data and provide methods related to them.
 * String is an object. Arrays are objects.
 *
 * The java.awt package provides the Point class (which is also an object)
 * intended to represent the coordinates of a location in a Cartesian plane.
 *
 * awt means "Abstract Window Toolkit".
 *
 * We should first import the java.awt.Point class to use it.
 */
```

```
import java.awt.*;
//Point and Rectangle
import java.io.*;
//PrintStream

public class Chap10 {

    public static void main(String[] args) {
        PrintStream out = System.out;
```

10.1 & 10.2 Point objects and attributes

```
out.println("10.1 & 10.2 Point objects and attributes");
//Create a new point with the new operator:
Point p = new Point(3, 4);
//The result of the new operator is a reference to the new object.

//Variables that belong to an object are usually called attributes,
//or fields.
//For example, a Point object have two attributes: int x and int y.
//To access an attribute of an object, Java uses dot notation:
int x = p.x;
//meaning int x is assigned the value of attribute x of the object p
//referred to.
```

10.1 & 10.2 Point objects and attributes

```
out.printf("The coordinate of blank is (%d, %d).\n", p.x, p.y);
double distance =
out.printf("The distance from blank to origin is %.1f.\n", distance);
out.println();
```

10.1 & 10.2 Point objects and attributes

```
out.printf("The coordinate of blank is (%d, %d).\n", p.x, p.y);
double distance = Math.sqrt(Math.pow(p.x, 2.0) + Math.pow(p.y, 2.0));
out.printf("The distance from blank to origin is %.1f.\n", distance);
out.println();
```

10.3 Objects as parameters

```
out.println("10.3 Objects as parameters");
//We can pass objects as parameters in the usual way:
printPoint(p);
//We can also:
out.println(p);
```

10.3 Objects as parameters

```
public static void printPoint(Point p) {


}
```

10.3 Objects as parameters

```
public static void printPoint(Point p) {
        System.out.printf("(%d, %d)\n", p.x, p.y);
}
```

10.3 Objects as parameters

```
//Rewrite the distance method from 6.2 so that it takes 2 Points as parameters
//instead of 4 doubles:
out.println(distance(p, new Point()));
//new Point() refers to (0, 0) by default.
out.println();
```

10.3 Objects as parameters

```
public static double distance(Point p1, Point p2) {


}
```

10.3 Objects as parameters

```
public static double distance(Point p1, Point p2) {
        return Math.sqrt(Math.pow(p2.x - p1.x, 2.0) + Math.pow(p2.y - p1.y, 2.0));
}
```

10.4 Rectangle object and Object as return types

```
out.println("10.4 Rectangle object and Object as return types");
//We should first import java.awt.Rectangle to use it.
//Rectangles have four attributes: int x, int y, int width and int height.
Rectangle box = new Rectangle(0, 0, 100, 200);
out.println(box);
//We can write methods that return objects.
out.println(center(box));
out.println();
```

10.4 Rectangle object and Object as return types

```
public static Point center(Rectangle box) {


}
```

10.4 Rectangle object and Object as return types

```
public static Point center(Rectangle box) {
        return new Point(box.x + box.width / 2, box.y + box.height / 2);
}
```

10.5 Mutable objects

```
out.println("10.5 Mutable objects");
//We can change the contents of an object by making an assignment to its
//attributes:
box.x = box.x + 50;
box.y = box.y + 100;
out.println(box);
```

10.5 Mutable objects

```
//The translate(int dx, int dy) method moves Rectangle upward by dx and
//rightward by dy.
box.translate(-50, -100);
out.println(box);
//This is a good illustration of object-oriented programming. Rather than
//write methods that modify one or more parameters, we apply methods to
//objects using dot notation. java.awt provides a number of methods that
//operate on Points and Rectangles.
out.println();
```

10.6 Aliasing

```
out.println("10.6 Aliasing");
//It is possible to have multiple variables referring to the same object.
Rectangle box1 = new Rectangle(0, 0, 100, 200);
Rectangle box2 = box1;
box1.translate(50, 50);
out.println(box1);
out.println(box2);
//It seems both box1 and box2 move upward by 50 and rightward by 50.
//Because they are referring to the same object. They are aliases.
out.println();
```

10.7 The null keyword

```
out.println("10.7 The null keyword");
//In java, the keyword null is a special value that means "no object".
Point blank = null;
//x = blank.x;
//blank.translate(50, 50);
//Either accessing an attribute of a null value or invoking a method
//will cause NullPointerException.
```

10.7 The null keyword

```
//It is legal to pass a null reference as an argument or receive one as
//a return value. Null is often used to represent a special condition
//or indicate an error.
out.println();
```

10.8 Garbage collection

```
out.println("10.8 Garbage collection");
//What happens when no variables refer to an object?
p = null;
//If there are no references to an object, there is no way to access its
//attributes or invoke a method on it. However it's still present in the
//computer's memory, taking up space.
//As the program runs, the system automatically looks for stranded objects
//and reclaims them. This process is called garbage collection.
out.println();
```

10.10 Java library source

```
out.println("10.10 Java library source");
//On OS X - /Library/Java/JavaVirtualMachines/jdk.../Contents/Home/
//On Windows - C:\Program Files\Java\jdk...
//On Linux - /usr/lib/jvm/openjdk-8/
//Find the src.zip/
out.println();
```