Solutions to questions of lecture 3
by Kasper B. Graversen

version 1 - 19/02-04

version 2 - 20/2
   fixed exercise 4.6

1.2
"hello dear chap"

1.3
"Yo man"

1.4
"Yo man"

1.5
As the error states, you have a clash of names which makes the class "greet.Greetings" ambigious.

1.6
Since GreetingTest is dependent on a Greeting, one must be accessible during compilation of GreetingTest.

Which actual Greeting is provided during compilation is crusial, since GreetingTest then relies on the interface of Greeting (i.e. which methods can be called on Greeting objects, what arguments its constructor requires etc). So when providing a different Greeting when running GreetingTest, one can create run-time errors since the provided Greeting class may be completely different than the Greeting class GreetingTester was compiled with.

2.1
```
import javax.swing.JFrame;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
public class DotFrame extends JFrame {

        public static void main(String[] args){
                final DotFrame df = new DotFrame();
                df.addMouseListener( df.new MakeDotOnMouseClick() );    // changed due to ex
1
        }

        // ex 1
        class MakeDotOnMouseClick extends MouseAdapter {
                public void mouseClicked(MouseEvent e) {
                        DotFrame.this.getGraphics().fillOval(e.getX()-5,e.getY()-5,10,10);
                }
        }

        public DotFrame() {
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                setSize(400, 400);
                setTitle("DotFrame");
                setVisible(true);
        }
}
```

2.2
```
import javax.swing.JFrame;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;

public class DotFrame extends JFrame {
```

```
        public static void main(String[] args){
                final DotFrame df = new DotFrame();
                df.addMouseListener( new MakeDotOnMouseClick(df) );     // changed due to ex
2
        }

        public DotFrame() {
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                setSize(400, 400);
                setTitle("DotFrame");
                setVisible(true);
        }
}


// ex 2
class MakeDotOnMouseClick extends MouseAdapter {
        DotFrame df;    // the instance on which we draw the dot...
        public MakeDotOnMouseClick(DotFrame df) {
                this.df = df;
        }

        public void mouseClicked(MouseEvent e) {
                df.getGraphics().fillOval(e.getX()-5,e.getY()-5,10,10);
        }
}
```

2.3
The last paragraph in 9.11 states that if a local class refers to variables or formal parameters in the enclosing method or constructor, those variables must be declared final. The variable df is declared in the enclosing main mathod of the local class, hence it must be final.

3.1
the main method has been declared static, hence it lives outside the scope of any objects of the class in which the main method resides. The field serialNo is not declared static, hence it is a description of how objects of the class the field resides in will be shaped---it is a field in every object created. The main method did not create any objects, hence it cannot access the serialNo field as it is non-existing.

3.3
We can access the nextSerialNumber field from within main since the field has been declared static, and hence live outside any objects of the class Bike. The main method and the field are in the same scope, hence they can always access each other.

3.7
Code in a static scope can only reach things in a static scope (since there is no guarantee any objects are created at that point in the execution). Code in a dynamic scope can access everything in a static scope as well as things in dynamic scope.. eg. itself. It can also access other things in a dynamic scope such as values in other objects (provided it has references to it).

code for ex 3

```
public class Bike {
        static int nextSerialNumber; // ex 3

        private String serialNo;                // of the form axb889<serialNo>
        private int size;                               // wheel size in inches
        private String color;

        public String getSerialNo(){ return serialNo; }
```

```
        public int getSize(){ return size; }
        public String getColor() { return color; }


        // ex 6
        public void setNextSerialNumber(int n) {
                Bike.nextSerialNumber = n;
        }


        // ex 2
        // public Bike(String serialNo, int size, String color) {
        //     this.serialNo = serialNo;
        //         this.size = size;
        //     this.color = color;
        // }

        // ex 5
        public Bike(int size, String color) {
                    this.serialNo = "axb889" + Bike.nextSerialNumber++;
                    this.size = size;
                    this.color = color;
        }

        // public static void main(String[] args){
        //      nextSerialNumber = 10001; // ex 3

                // ex 2
        //      Bike lightning = new Bike("axb88910001", 27, "Red");
        //      System.out.println("my super bike is: " + lightning.getSerialNo() + " size "
+ lightning.getSize() + " in " + lightning.getColor());
        // }

}


// ex 6
class BikeTest {
        public static void main(String[] args){
                Bike.setNextSerialNumber(10001);
                Bike lightning = new Bike(27, "Red");
                System.out.println("my super bike is: " + lightning.getSerialNo() + " size "
+ lightning.getSize() + " in " + lightning.getColor());
        }
}
```