

## Exercise 10.1

/\*\* Saket Bakshi. 3/4/19. Period 6. This is used for question 1 of Chapter 10.

Adds a method to the Data class that returns the object with the greatest measure.

```
*/
public class Data
{
    /**
     * Computes the average of the measures of the given objects.
     * @param objects an array of Measurable objects
     * @return the average of the measures
     */
    public static double average(Measurable[] objects)
    {
        double sum = 0;
        for(Measurable obj : objects)
        {
            sum = sum + obj.getMeasure();
        }
        if(objects.length > 0) {return sum / objects.length;}
        else {return 0;}
    }

    /**
     * Finds the maximum an object has out of an array of given objects.
     * @param objects an array of Measurable objects
     * @return the object with the maximum
     */
    public static Measurable max(Measurable[] objects)
    {
        double max = 0;
        Measurable maximum = null;
        for(Measurable obj : objects)
        {
            if(obj.getMeasure() > max)
            {
                max = obj.getMeasure();
                maximum = obj;
            }
        }

        return maximum;
    }
}
```

```
/** Saket Bakshi. 3/4/19. Period 6. This is used for question 1 of Chapter 10.  
    Creates a Measurable interface to be used to obtain measures.  
*/
```

```
public interface Measurable  
{  
    /**  
        obtains a measure of a Measurable object  
    */  
    double getMeasure();  
}
```

```
PS C:\Users\saket\Git\CSWork\JAVA> java Printer  
No output is expected because a method has been added to the Data class.  
PS C:\Users\saket\Git\CSWork\JAVA>
```

## Exercise 10.2

/\*\* Saket Bakshi. 3/4/19. Period 6. This is used for question 2 of Chapter 10.

Creates a Quiz class that implements the interface Measurable so we can test interfaces.

\*/

```
public class Quiz implements Measurable
{
```

```
    private int score;
    private String letterGrade;
```

```
    /**
```

```
        Constructs an object with a score and letter grade.
```

```
    */
```

```
    public Quiz()
```

```
    {
```

```
        score = 0;
        letterGrade = "";
```

```
    }
```

```
    /**
```

```
        Constructs an object with a score and letter grade.
```

```
        @param scored the score
```

```
        @param grade the letter grade
```

```
    */
```

```
    public Quiz(int scored, String grade)
```

```
    {
```

```
        score = scored;
        letterGrade = grade;
```

```
    }
```

```
    /**
```

```
        Returns the score.
```

```
        @return the score
```

```
    */
```

```
    public double getMeasure()
```

```
    {
```

```
        return score;
```

```
    }
```

```
    /**
```

```
        Returns the grade
```

```
        @return the letter grade
```

```
    */
```

```

        public String getGrade()
        {
            return letterGrade;
        }
    }
    /** Saket Bakshi. 3/4/19. Period 6. This is used for question 2 of Chapter 10.
        Tests a Quiz class that implements the interface Measurable so we can test interfaces.
    */
    public class QuizTester
    {
        public static void main(String[] args) {
            Quiz[] scores = new Quiz[3]; //makes array of Quiz objects
            scores[0] = new Quiz(100, "A+");
            scores[1] = new Quiz(40, "F");
            scores[2] = new Quiz(70, "C-");

            double averageScore = Data.average(scores); //uses Data class methods
            System.out.println("Average score: " + averageScore);
            System.out.println("Expected average: 70");

            Quiz bestScore = (Quiz)Data.max(scores); //assigns best quiz grade to a new
object

            int maxScore = (int)bestScore.getMeasure();
            String maxGrade = bestScore.getGrade();

            System.out.println("The maximum score was " + maxScore + ". This score
received a grade of " + maxGrade + ".");
        }
    }
}

```

```

PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C10EXBakshiSaket> java QuizTester
Average score: 70.0
Expected average: 70
The maximum score was 100. This score received a grade of A+.
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C10EXBakshiSaket>

```

### Exercise 10.3

/\*\* Saket Bakshi. 3/4/19. Period 6. This is used for question 3 of Chapter 10.

Creates a Person class that implements the interface Measurable so we can test interfaces.

\*/

public class Person implements Measurable

{

private int height;

private String name;

/\*\*

Constructs an object with a name and height.

\*/

public Person()

{

height = 0;

name = "";

}

/\*\*

Constructs an object with a name and height.

@param centimeters the height of the person

@param nombre the name of the person

\*/

public Person(int centimeters, String nombre)

{

height = centimeters;

name = nombre;

}

/\*\*

Returns the height.

@return the height

\*/

public double getMeasure()

{

return height;

}

/\*\*

Returns the name

@return the name

\*/

```

        public String getName()
        {
            return name;
        }
    }

```

/\*\* Saket Bakshi. 3/4/19. Period 6. This is used for question 3 of Chapter 10.

Tests a Person class that implements the interface Measurable so we can test interfaces.

\*/

```

public class PersonTester
{

```

```

    public static void main(String[] args) {
        Person[] people = new Person[3];
        people[0] = new Person(180, "Saket");
        people[1] = new Person(140, "Billy");
        people[2] = new Person(190, "Jack");

        double averageHeight = Data.average(people);
        System.out.println("Average score: " + averageHeight);
        System.out.println("Expected average: 170");

```

```

        Person tallest = (Person)Data.max(people);

```

```

        int maxHeight = (int)tallest.getMeasure();
        String nombre = tallest.getName();

```

```

        System.out.println("The maximum height was " + maxHeight + ". This height was
on " + nombre + ".");
    }
}

```

```

PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C10EXBakshiSaket> java PersonTester
Average score: 170.0
Expected average: 170
The maximum height was 190. This height was on Jack.
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C10EXBakshiSaket>

```

#### Project 10.14

/\*\* Saket Bakshi. 3/4/19. Period 6. This is used for question 1 of Chapter 10.

Creates a PopulationGrowth class that uses a button to simulate the growth of a cockroach population.

\*/

```
public class PopulationGrowth
```

```
{
```

```
    private int population;
```

```
    /**
```

```
        Constructs an empty population.
```

```
    */
```

```
    public PopulationGrowth()
```

```
    {
```

```
        population = 0;
```

```
    }
```

```
    /**
```

```
        Constructs a population with a given size.
```

```
        @param size the initial size of the population.
```

```
    */
```

```
    public PopulationGrowth(int size)
```

```
    {
```

```
        population = size;
```

```
    }
```

```
    /**
```

```
        Doubles the population.
```

```
    */
```

```
    public void doubler()
```

```
    {
```

```
        population = population + population;
```

```
    }
```

```
    /**
```

```
        Returns the population.
```

```
        @return the population
```

```
    */
```

```
    public int getPop()
```

```
    {
```

```
        return population;
```

```
    }
```

```
}
```

/\*\* Saket Bakshi. 3/4/19. Period 6. This is used for question 1 of Chapter 10.

Tests a PopulationGrowth class that uses a button to simulate the growth of a cockroach population.

\*/

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JTextField;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

public class PopulationGrowthTester

{

private static final int FRAME\_WID = 400; //frame dimensions

private static final int FRAME\_HEI = 100;

public static void main(String[] args) {

JFrame frame = new JFrame();

JButton button = new JButton("Simulate population increase"); //name of the  
button

final PopulationGrowth cockroaches = new PopulationGrowth(2); //creates an  
initial population of 2

final JLabel label = new JLabel("population: " + cockroaches.getPop()); //creates  
a label to show population

JPanel panel = new JPanel();

panel.add(button);

panel.add(label);

frame.add(panel);

class PopulationDoubleListener implements ActionListener

{

public void actionPerformed(ActionEvent event)

{

cockroaches.doubler(); //doubles population

label.setText("population: " + cockroaches.getPop()); //updates the  
label

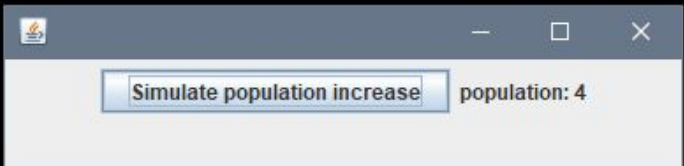
}

}



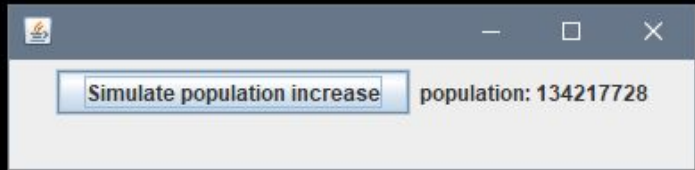
```
        ActionListener listener = new PopulationDoubleListener(); //creates a listener  
        button.addActionListener(listener); //adds the listener to the button  
        frame.setSize(FRAME_WID, FRAME_HEI);  
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        frame.setVisible(true);  
    }  
}
```

```
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C10EXBakshiSaket> java PopulationGrowthTester
```



The screenshot shows a Java Swing window with a title bar containing a standard icon, a minus sign, a maximize button, and a close button. The window's content area has a light gray background. On the left side, there is a button with the text "Simulate population increase". To the right of the button, the text "population: 4" is displayed.

```
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C10EXBakshiSaket> java PopulationGrowthTester
```



The screenshot shows the same Java Swing window as above, but the text to the right of the button now reads "population: 134217728", indicating that the population has been updated after clicking the button.