COMP601 - ASSIGNMENT TWO

AIMS

The course work tasks aim to practice the following topics:

- Basic input and output operations
- Coding standards
- Debugging and testing by using features of Integrated Development Environment (IDE)
- The use of methods (both static and instance) and parameters in solving problems
- Design and implement classes and objects
  - public and private class fields
  - constructors, public, and private methods
  - Modular programming
  - Problem solving using classes and objects
- Derived classes and protected variables and methods: Examples and problem solving
- Polymorphism and its advantages in software development: Examples and problem solving
- Using object oriented programming to develop applications
- Object oriented programming and code reuse
- Array of objects
- Object persistence with file I/O

MARKING PROCESS

STEP 1

Copy all your ***.java*** files into a submission folder (you don't have to submit everything in Eclipse project directory, just the .java file under the **src** folder), **right-click** the submission folder, ***Send to***, ***Compressed zipped folder***. A zip file should be created.

Upload the **zip file** to Moodle under the assessments tab before in-class demonstration (see timetable on Moodle for due date).

STEP 2

Demonstrate your solution to your tutor during class. You may have to explain to your tutor how your program works, and be prepared to answer questions with regards to your solution.

MARKING CRITERIA AND MARK ALLOCATION

Assignment Two is worth 25% of final grade. To achieve the indicated mark on each question, your solution must be complete and function correctly according to both of the following:

- The question requirement description.
- The sample run outputs (if any).
- Your program must follow the **Programming Best Practices** – you can find the document on Moodle.

Question #	1	2	3	4	5	Total
Marks	15	15	20	15	35	100

Note, there is a marking guide for each question at the end of this document.

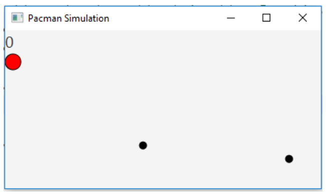
QUESTION 1 – JAVAFX GUI APPLICATION

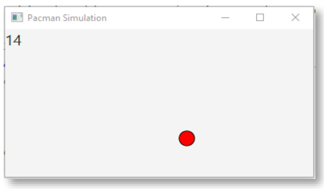
- Learning outcomes: 1, 2, 3
- Relevant topics: input, output, iteration, array, class, objects, method invocation
- Resources: week 4 videos, in-class examples, reference material on Moodle, recommended textbooks
- Moodle section reference: JavaFX GUI

To complete this task, first complete JavaFX tutorial (Part 1-3) on Moodle.

CAUTION: do not create *Java Project*, what you need is **JavaFX Project (File, New, Other…, JavaFX, JavaFX Project)**.

At the end of the tutorial, your program should have a game interface similar to the following screenshots. Screenshot on the left shows game start, the screenshot on the right shows the timer stopped at 14 seconds after all black dots were removed.

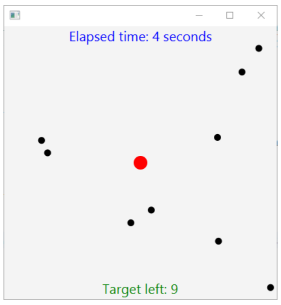


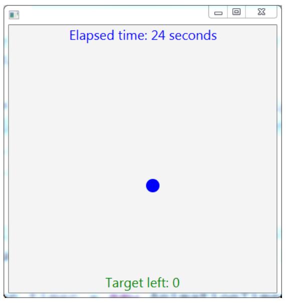


Your task is to modify the program to add the following features. Each feature is worth a number of marks (see marking guide at the end of the question). The screenshots further below give you the idea of what the features look like. The four features are described as below:

1. Modify the timer label text to make it more descriptive (rather than just a number).
2. When game is over, the Pacman (the moving red circle) should change to a colour of your choice (eg, blue).
3. The number of black dots should be a random number between 5-10 (inclusive), i.e., each time the game is run, there should be a different number of black dots. Hint: use a loop whose iteration is controlled by the generated random number.
   Note, please do not confuse this random number with the <u>random placement of each black dot</u>. The two variables in the tutorial (randomx and randomy) are used to randomly place each dot on the canvas. You need to create a separate random number which controls <u>how many dots</u> are to be generated and placed on the canvas.
4. Add another label showing the number of remaining black dots. The displayed remaining number should be updated along with each black dot removed.
   - First, set the random number generated in feature 3 as the initial value on the label. CAUTION: do NOT do this inside the **handle**() method of the AnimationTimer block; otherwise, updating the label later on won't work. This is because the handle() method runs every second and the label gets reset to its initial value each time.
   - Then, as each black dot is removed, use "allCircles.size()-1" to retrieve the remaining number of black dots, and set the value to the "target-left" label. The sample run screenshot adds the label at the bottom-centre of the canvas – you could also set it elsewhere as you see fit.

The following are sample run screenshots showing the start and end of game with the new features:





QUESTION 2 – ENCAPSULATION

- Learning outcomes: 1, 2, 3
- Relevant topics: input, output, iteration, array, class, objects, encapsulation
- Resources: week 4, 5, 6 videos, in-class examples, reference material on Moodle, recommended textbooks
- Moodle section reference: Encapsulation

Create a class Note, as in note in music. A Note object is represented by a pitch (one of the seven letters – C, D, E, F, G, A, and B – which is the simplest and most common way of specifying a music note), and a duration (in number of seconds). Write the class Note that contains

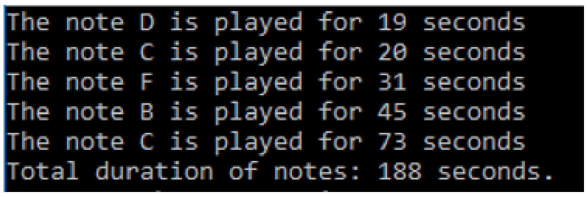
- Private class fields with getter/setter: pitch (String), duration (int).
- A parameterized constructor that takes two parameters and sets the fields.
- A public method playNote() that outputs the pitch and the duration of a note in the format as in the sample output below (i.e., "The note <pitch letter> is played for <duration> seconds"). Method header is given as following: `public void playNote()`

The code in `main()` is given below. The code uses the array shorthand notation syntax to create an array of Note objects. The array of objects is then passed to the method `processNotes`.

```
public class Main {
        public static void main(String[] args){
                Note[] noteArray = new Note[5];
                noteArray[0] = new Note("D", 19);
                noteArray[1] = new Note("C", 20);
                noteArray[2] = new Note("F", 31);
                noteArray[3] = new Note("B", 45);
                noteArray[4] = new Note("C", 73);

                processNotes(noteArray);
        }//end of main method
        public static void processNotes(Note[] notes){
             //Write code to produce the sample run output below
             //Hint: use a loop to iterate "notes" array, and invoke playNote()
             //on each object.
             //Calculate the total duration of all Note objects in the loop as well.
         }
}
```

Write the code for the processNotes method. When run, the program should produce an output similar to the following:



```
The note D is played for 19 seconds
The note C is played for 20 seconds
The note F is played for 31 seconds
The note B is played for 45 seconds
The note C is played for 73 seconds
Total duration of notes: 188 seconds.
```

QUESTION 3 – INHERITANCE

- Learning outcomes: 1, 2, 3
- Relevant topics: input, output, iteration, array, class, objects, encapsulation, inheritance
- Resources: week 5 & 6 videos, in-class examples, reference material on Moodle, recommended textbooks
- Moodle section reference: Inheritance

Create a class Shape, that contains:

- Private field with getter/setter: colour (String).
- A parameterized constructor that takes a single parameter and sets the field.
- A method getShapeType() that returns the message "I'm a Shape".

  Method header: `public String getShapeName()`

Create a class Rectangle, as subclass of Shape. It contains:

- Private fields with getter/setter: height (double), and width (double).
- A parameterized constructor that takes three parameters. Call superclass constructor using the "super" keyword.
- An override method getShapeName() that returns the message "I'm a Rectangle".

  Method header: `@Override public String getShapeName()`
- A method getPerimeter() that calculates and returns the rectangle perimeter.

  Method header: `public double getPerimeter()`

In the main() method below, two arrays of objects are created. Write the rest of code to produce the sample run output:

```
public class Main {
    public static void main(String[] args) {
        List<Shape> shapes = new LinkedList<>();
        shapes.add(new Shape("white"));
        shapes.add(new Rectangle("red", 10, 6));
        shapes.add(new Rectangle("black", 20, 9));
        shapes.add(new Shape("orange"));
        showShapeNames(shapes);//produces output part 1


        Rectangle[] rectangleArray1 = {
                                new Rectangle("white", 4, 3),
                                new Rectangle("red", 9, 5),
                                new Rectangle("purple", 3, 6),
                                new Rectangle("orange", 15, 10),
                                new Rectangle("black", 4, 14),
                            };
        Rectangle[] rectangleArray2 = {
                                new Rectangle("pink", 3, 4),
                                new Rectangle("red", 10, 2),
                                new Rectangle("white", 8, 5),
                                new Rectangle("blue", 14, 4),
                                new Rectangle("bindle", 10, 15),
                            };
        //produces output part 2
        countOverlapRectangles(rectangleArray1, rectangleArray2);
    }//end of main() method
    public static void showShapeNames(List<Shape> shapes){
        //Complete method to produce sample run output part 1
        //Hint: iterate list "shapes", invoke getShapeName() on each instance

    }
    public static void countOverlapRectangles(Rectangle[] group1, Rectangle[]
   group2){
        // Complete method to produce sample run output part 2
        // Note: you can assume that the colours in each group are unique,
        // i.e., there are no duplicate colours within each group.
    }
}//end of Main class
```

Sample run output part 1:

```
I'm a Shape
I'm a Rectangle
I'm a Rectangle
I'm a Shape
```

Sample run output part 2:

```
There are 2 Rectangle objects with overlapping colour between the two arrays
There are 3 Rectangle objects with overlapping perimeter between the two arrays
```

QUESTION 4 – RECURSION OF A LIST OF OBJECTS

- Learning outcomes: 1, 2, 3
- Relevant topics: input, output, recursion, methods, class, objects
- Resources: week 9 videos, in-class examples, reference material on Moodle, recommended textbooks
- Moodle section reference: Recursion

Create a class Book that contains

- Private fields with getter/setter: title (string), numOfPages (int)
- A parameterized constructor with two parameters

Use the code given below in the main(), which creates a list of objects; the list is then passed to the method calcTotalPages.

```
public class Main {
     public static void main(String[] args)    {
         List<Book> books = new LinkedList<>();

         //fill in code here: create a list of Book instances
         //and add to the list “books”.

         BookApp bkapp = new BookApp();
         int totalPages = bkapp.calcTotalPages(books, 0);
         out.printf(“Total number of books: %d\n”, books.size())
         out.printf(“Total pages: %d\n”, totalPages);
     }
}//end of Main class

public class BookApp {
     pubilc int calcTotalPages(List<Book> bookList, int accum)  {
         //Write the code for this recursive method
         //This calculates and returns the total number of pages
         //of all instances in “bookList”.
     }
}//End of class BookApp
```

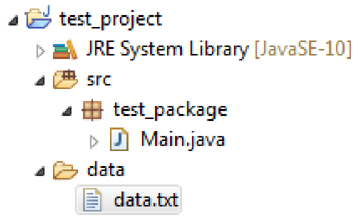
A sample run is below:
Total number of books: 3
Total pages: 1560

QUESTION 5 – OBJECTS AND FILE I/O

- Learning outcomes: 1, 2, 3
- Relevant topics: input, output, class, objects, methods, file input/output
- Resources: week 10 videos, in-class examples, reference material on Moodle, recommended textbooks
- Moodle section reference: File I/O

Create an Eclipse project, then create a folder named data at the top level of the project. To create a folder in Eclipse, in the Package Explorer pane, right-click project name, New, Folder, then type the folder name **data** in the popup dialog. To create a text file in the data folder, right-click the folder name, New, File, then type **data.txt** in the popup dialog.

Your project structure should look similar to the following screenshot:



Create the text file **data.txt** with the content listed in the box to the right. The text contains information about five student records, with <student_id> and <score> separated by comma, one record per line.

```
L0111,99500
L0222,75200
L0333,65400
L0444,67900
L0555,56500
```

Create a class Employee, which contains:

- Private class fields with getter/setter: ID (String), salary (double)
- A parameterized constructor with two parameters
- A void method displayEmployee() that displays ID and salary.

Create a class SalaryApp, which contains:

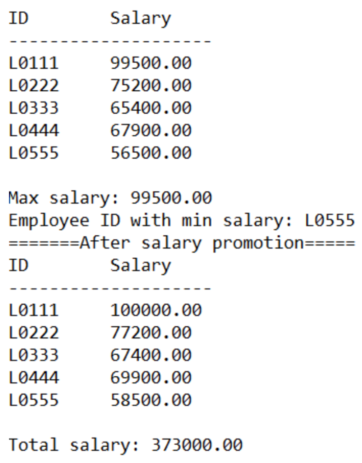
- Create a list of Employee objects **people** as a class field, with getter.
- Write a paramerised constructor, with a file name as its parameter.
- Create a method **readData()** to read the file content and populate the objects. In particular, first initialize the **people** list, then read the file content, line by line, until the end of the file. Split each line into two items (use the Split method); use the two items to create a **Employee** object and add the object to the list.
- Write a method **printAll()**, in which, traverse through the list and output information of each Employee object to screen.
- Write a method **getMaxSalary()** that calculates and returns the highest salary
- Write a method **getMinSalaryID()** that calculates and returns the ID of the Employee who earns the lowest salary.
- A method **promoteSalary()** that iterates through the array of object and gives each Employee a $2000 pay rise. Hint: apply the method setSalary() on each Employee instance. Note: employee salary must be capped at $100000, i.e., if a person's salary goes over $100000 after the $2000 promotion, his/her salary is set to $100000.
- In the Main class, write a **static recursive** method. The method recursively calculates the total salary of all employees. The method should return the total in the end.

The following is a suggested template:

```java
public class Main {

  public static void main(String[] args) {
    String txtfile = "data/data.txt";
    try {
        SalaryApp salaryapp = new SalaryApp(txtfile);
        app.printAll();
        System.out.println();
        out.printf("Max salary: %.2f\n", salaryapp.getMaxSalary());
        out.printf("Min salary employee ID: %s\n", salaryapp.getMinSalaryID());
        salaryapp.promoteSalary();
        System.out.println("=======After salary promotion=====");
        salaryapp.printAll();
        System.out.println();
        double total = recursiveTotalSalary(salaryapp.getPeople(), 0);
        out.println("Total salary: %.2f\n", total);
    }catch(IOException ioe) {
        out.printf("Perhaps missing text file: %s/%s? \n\n",
                     new Main().getClass().getPackage().getName(), txtfile);
    }
  }
  public static double recursiveTotalSalary(List<Employee> list, double accum) {
       //write code for this recursive method
  }
}//End of class Main
public class SalaryApp{
       private List<Employee> people;
       public List<Employee> getPeople() { return people; }
       public SalaryApp(String filename) throws IOException {
           people = new LinkedList<>();
           readData(filename);
       }
       public void readData(String fn) throws IOException {
              // write the code
       }
       public void printAll() {
              // write the code
       }
       public String getMinSalaryID() {
              // write the code
       }
      public double getMaxSalary() {
              // write the code
       }
       public void promoteSalary() {
              // write the code
       }
}//End of class SalaryApp
public class Employee {
       // write the class code
}//End of class Employee
```

Here is a sample screenshot:



```
ID        Salary
--------------------
L0111     99500.00
L0222     75200.00
L0333     65400.00
L0444     67900.00
L0555     56500.00

Max salary: 99500.00
Employee ID with min salary: L0555
=======After salary promotion=====
ID        Salary
--------------------
L0111     100000.00
L0222     77200.00
L0333     67400.00
L0444     69900.00
L0555     58500.00

Total salary: 373000.00
```

Name: ☐ **I pledge by honour this is solely my own work** Score:

☐ Work submitted on Moodle

ASSIGNMENT TWO MARKING GUIDE

QUESTION 1 MARKING GUIDE	Total: 15
Feature 1: more descriptive timer label text	2
Feature 2: Pacman changes colour at gameover	2
Feature 3: generate a random number of target dots	5
Feature 4: add a label showing remaining number of targets and it works according to requirement	5
Programming Best Practices	1

QUESTION 2 MARKING GUIDE	Total: 15
Class Note: fields (with getter/setter), constructor, method displayNote	7
Method processHand: iterating the array “notes” and invoke the displayNote() method on each object	7
Programming Best Practices	1

QUESTION 3 MARKING GUIDE	Total: 20
Class Shape: fields (with getter/setter), constructor. method getShapeName	4
Class Rectangle: fields (with getter/setter), constructor Method getShapeName Method getPerimeter	5
Method showShapeNames	5
Method countOverlapRectangles	5
Programming Best Practices	1

QUESTION 4 MARKING GUIDE	Total: 15
Book class: fields with getter/setter, constructor	5
Solution utilises tail recursion technique	2
Correctly implement recursive and base case	7
Programming Best Practices	1

QUESTION 5 MARKING GUIDE	Total: 35
Employee class, with fields and getter/setter, and displayEmployee method	3
Method readData() Initialize the people list Read the file content, line by line, until the end of the file. Split each line into two items (use the Split method) Use the two items to create a Employee object and add the object to the people list	5
Method printAll() that traverses the list and outputs each object info to screen	3
Method getMinSalaryID() that calculates and returns the employee ID who has the minimum salary.	5
Method getMaxSalary() that calculates and returns the maximum salary.	5
Method promoteSalary() that iterates through the array of object and gives each Employee a $2000 pay rise.	6
In the Main class, write a static recursive method. The method recursively calculates the total score of all students. The method should return the total in the end.	7
Programming Best Practices	1