BIRZEIT UNIVERSITY

Computer Science Department
COMP2311 Mid Term Exam
Time: 80 minutes

Student Name: _____KEY_____ Student ID#: _____

Choose your instructor:

Dr. Mamoun Nawahdah	Dr. Murad Njoum	Dr. Abdallah Karakra
Lab 3 (S, 08:00 - 10:40) ☐	Lab 1 (M, 14:15 - 16:55) ☐ Lab 2 (T, 11:25 - 14:05) ☐	Lab 4 (M, 14:15 - 16:55) ■ Lab 5 (T, 08:00 - 10:40) ■

Question	Q1	Q2	Q3	Total
Mark	20+15 /35	20 + 15 /35	30 /30	100

Question I (35%)

Part A (20%)

Select the best answer for each of the following questions (1-10) and put your answers in the given table at page 4.

1. We cannot create a subclass of _____class.

A. abstract.
B. public.
C. static.
D. final.

2. What is an immutable object?

A. An immutable object can be changed once it is created.
B. An immutable object can't be changed once it is created.
C. An immutable object is an instance of a public class.
D. None of the above.

3. Which of the following statements about inheritance is correct?

A. You can always use a superclass object in place of a subclass object.
B. You can always use a subclass object in place of a superclass object.
C. A superclass inherits data and behavior from a subclass.
D. A superclass inherits only behavior from a subclass.

4. What is the output of the following code segment?

```
try {
        System.out.print("Hello World");
    } catch (Exception e) {
        System.out.println("e");
    } catch (ArithmeticException e) {
        System.out.println("e");
    } finally {
        System.out.println("!");
    }
```

A. Hello World
B. Hello World!
C. It will throw runtime exception
D. It will not compile because the second catch statement is unreachable

5. To keep the circle at the <u>center of the pane</u> regardless of how the stage is resized (width or height). Among the following methods, <u>which is correct</u>?

A)
```
circle.centerXProperty().bind(pane.widthProperty().divide(2));
circle.centerYProperty().bind(pane.heightProperty().divide(2));
```

B)
```
circle.SetCenterXProperty().bind(pane.widthProperty().divide(2));
circle.SetCenterYProperty().bind(pane.heightProperty().divide(2));
```

C)
```
circle.SetCenterX(pane.widthProperty().divide(2));
circle.SetCenterY(pane.heightProperty().divide(2));
```

D)
```
No of the above
```

6. To write to a byte stream (Binary), you should use a class that extends which of the following abstract classes ?

A. InputStream
B. OutputStream
C. Reader
D. Writer

7. Consider the following code snippet:

```
public class Coin {
   private String name;
   . . .
   public boolean equals(Object otherCoin){
      return name.equals(otherCoin.name);
   }
   . . .
}
```

What is wrong with this code?

A. The return statement should use the == operator instead of the equals method.
B. The parameter in the equals method should be declared as Coin otherCoin.
C. otherCoin must be cast as a Coin object before using the equals method.
D. There is nothing wrong with this code

8. Given the declaration `Circle[] x = new Circle[10]`, which of the following statement is most accurate.

A. x contains a reference to an array and each element in the array can hold a reference to a Circle object.
B. x contains a reference to an array and each element in the array can hold a Circle object.
C. x contains an array of ten objects of the Circle type.
D. x contains an array of ten int values.

9. Which corner of the screen has the pixel coordinates (0, 0)?

A. Top-left
B. Top-right
C. Bottom-left
D. Bottom-right

10. Assume Book is an interface, and both "Dictionary" and "Encyclopedia" classes implement it. Which of the following statements is valid ?

A. `Book b = new Book();`
B. `Book d = new Dictionary();`
C. `Encyclopedia e = new Book();`
D. `none of the above`

Answer: Sheet for Question **I (Part A):املأ الجدول التالي باستخدام الأحرف الكبيرة فقط**

1	2	3	4	5	6	7	8	9	10
D	B	B	D	A	B	C	A	A	B

Part B (15%)

There are some errors with the codes from A to C. You should rewrite these codes to address those errors.

A) Assuming that the **Animal class** does define **a public eat method**.

```
Object animal = new Animal();
animal.eat();
```

/*the compiler complains because the `Object` class does not define an `eat` method.*/ 1pt

```
Object animal = new Animal();
((Animal)animal).eat();// 2pts
```

B)

```
public class Test {
    public static void main(String[] args) {
      java.util.Date x = new java.util.Date();
      java.util.Date y = x.clone();
      System.out.print("Hello World").
    }
}
```

```
public class Test {
  public static void main(String[] args) {
    java.util.Date x = new java.util.Date();
    java.util.Date y = (java.util.Date)x.clone();     // Cast   3pts
    System.out.print("Hello World");   //  ;   2pts
  }
}
```

C)

```
public class Test {
  public static void main(String[] args) {
    Person[] persons = {new Person(3), new Person(4), new Person(1)};
    java.util.Arrays.sort(persons);
  }
}

class Person {
     private int id;

     Person(int id) {
        this.id = id;
     }
}
```

```
public class Test {
   public static void main(String[] args) {
      Person[] persons = {new Person(3), new Person(4), new Person(1)};
     java.util.Arrays.sort(persons);
   }
}

class Person implements Comparable<Person>{ //  implements Comparable<Person>    2pts
     private int id;

     Person(int id) {
       this.id = id;
     }

    @Override
    public int compareTo(Person o) {          // method signature  2pts
       return (this.id- o.id);                //  override compareTo    3Pts

   }
}
```

Question II (35%)

Part A (20%)

True/False Questions: put your answers in the given table below.

		TRUE	FALSE
1.	Every JavaFX program is defined in a class that extends the interface Application.		**False**
2.	A particular **catch** block can catch exceptions from more than one **try** block.		**False**
3.	An interface has methods but no instance variables.	**True**	
4.	A "has-a" relationship is implemented via inheritance.		**False**
5.	**Exception** is a subclass of **Error**.		**False**
6.	Interfaces have both private and public methods.		**False**
7.	It is not possible to add a Shape or an ImageView directly to a Scene.	**True**	
8.	The following statements will create three objects **Student studentName, studentId, stud_class;**		**False**
9.	You can use the following statement to create a Color object: `new Color(1.2, 2.3, 3.5, 4);`		**False**
10.	A subclass inherits methods from its superclass but not instance variables.		**False**

Part B (15%)

Answer the following questions (A-C).

A) Suppose your code fills a **personList**[] array with objects instantiated from several **different classes derived (extended) from the Person class**, like **Doctor**, **Student**, **Driver**, etc. Write a Java statement that prints, "Tries to explain meaning of life" if and only if **personList [i]** refers to a Doctor.

```
if (personList [i] instanceof Doctor){                    // 3 pts
    System.out.println("Tries to explain meaning of life"); // 1 pt
}
```

B) Assume an **Animal class** defines a **public eat method** and **a Dog class derived (extended) from the Animal class** defines a different **public eat method**. Assume the declaration:

```
Animal[] animals =  {new Animal(), new Dog("Leo", "brown")};
```

Indicate which eat method is invoked by each of the following statements, and explain why.

```
animals[0].eat();
animals[1].eat();
```

The animal[0].eat(); statement invokes Animal's eat method // **2pts**
The animal[1].eat(); statement invokes Dog's eat method. // **2pts**

Reason //**2pts**
In each case, the JVM determines the type of the object referenced, and then it **binds** the method call to the method defined in that object's class.

C)
```
public class Mystery {
     static int a= 0;
     int b;
     public Mystery () {
          b= a;
          a= a+1;
      }
     public boolean equals (Mystery that) {
         return b == that.b;
     }
}
```

Is the result of (`new Mystery (). equals (new Mystery ())`**) <u>true</u>? <u>Why</u>?**

No // **2pts**

Reason // **3pts**

```
new Mystery (). equals (new Mystery ())
```

value of b=0 value of b=1

it will evaluate to true only if two Mystery objects are constructed at the same time, where here is not the case.

Question III (30%)

Write a class that implements the **Queue interface** called **MyQueue**, as shown in Figure 1 below. A queue is a data structure that accepts data and then returns it in the order in which it was received (**first-in, first-out order**). Items are added to the tail (نهاية) of the queue and removed from the head (بداية). See Figure 3.

```
public interface Queue {

  public int size(); //Returns number of objects in queue

  public boolean isEmpty(); //Returns true if queue is empty

  /* Adds an item to the tail of the queue */
  public void addLast(Object o);

  /*Removes and returns the item from the head of the queue */
  public Object removeFirst();
}
```

Figure 1: Queue interface

Your queue implementation must be **accessible and usable** from **any package**. However, any **attempt to extend your class should produce a compile-time error (your class must be protected in such a way that it cannot be extended by others).** Figure 2 illustrates a **sample main method** and **sample output**.

Sample main method

```
public static void main(String[] args) {
   Queue line = new MyQueue();
   line.addLast("Hello");
   line.addLast("World");
   System.out.println(line.removeFirst());
   System.out.println(line.removeFirst());
}
```

Output

```
Hello
World
```

Figure 2: sample main method and sample output

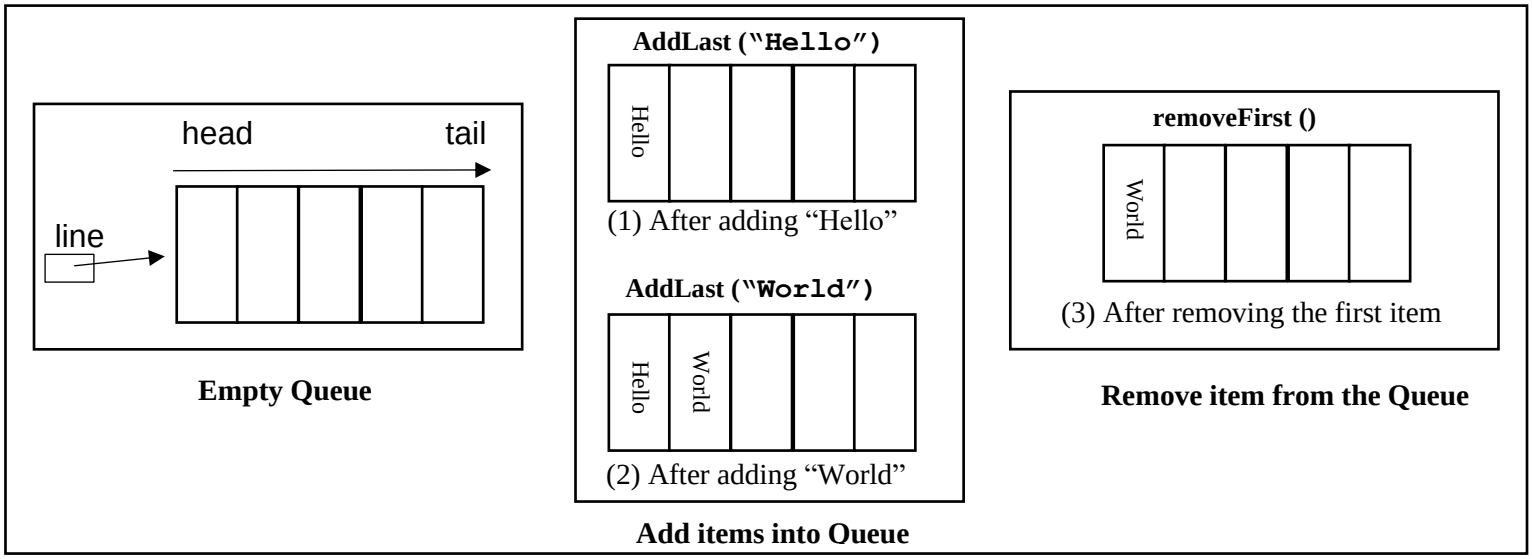

Figure 3: Queue implemented using ArrayList

Use Only ArrayList for storing(تخزين) the items

```
import java.util.ArrayList;                    // 2pts  for the import statement

public final class MyQueue implements Queue {   // 2pts for the final keyword ,   2 pts for the implements keyword

   private ArrayList list = new ArrayList();  // 2pts for the creation of  ArrayList , 2pts for the private keyword

   public MyQueue() {           // 2pts for the constructor

   }

   public int size() {           //  2 pts for correct signature

       return list.size();   // 2 pts
  }

  public boolean isEmpty() {  //  2 pts for correct signature

       return (list.size() == 0);  // 2 pts
  }

 public void addLast (Object o) {  //  2 pts for correct signature

    list.add(o);  // 2 pts

 }

 public Object removeFirst() {   //  2 pts for correct signature
   if (!isEmpty())   // 1 pts
       return list.remove(0);  // 2 pts

   return null; // 1pt
 }


}
```

Good Luck