INTERNATIONAL CURRICULUM CENTRE

RENDA FUZHONG

IBDP Computer Science Major Assessment 3
Option: Java OOP

Name: Marks:

Total questions: 11， Total Marks: 39， Exam Time: **80** minutes

Part1 is MCQ please answer write your answers in the table.

Part 2 is free-response questions. Please answer the questions on paper.

We will collect the written paper in **60** minutes

Part 3 is programming test in our wiki homepage, please submit the code to your project page in **20** minutes.

Part 1： MCQ --- 9 marks

1	2	3	4	5	6	7	8	9

1.

Suppose the characters 0, 1, . . . , 8, 9, A, B, C, D, E, F are used to represent a hexadecimal (base-16) number. Here A = 10, B = 11, . . . , F = 15. What is the largest base-10 integer that can be represented with a two-digit hexadecimal number, such as 14 or 3A?

(A) 32
(B) 225
(C) 255
(D) 256
(E) 272

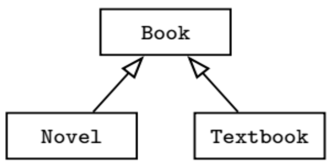
2

The color of a pixel can be represented using the RGB (Red, Green, Blue) color model, which stores values for red, green, and blue, each ranging from 0 to 255. How many bits (binary digits) would be needed to represent a color in the RGB model?

(A) 8
(B) 16
(C) 24
(D) 32
(E) 40

3

Consider this inheritance hierarchy, in which Novel and Textbook are subclasses of Book.



Which of the following is a *false* statement about the classes shown?

(A) The Textbook class can have private instance variables that are in neither Book nor Novel.
(B) Each of the classes—Book, Novel, and Textbook—can have a method computeShelfLife, whose code in Book and Novel is identical, but different from the code in Textbook.
(C) If the Book class has private instance variables title and author, then Novel and Textbook cannot directly access them.
(D) Both Novel and Textbook inherit the constructors in Book.
(E) If the Book class has a private method called readFile, this method may not be accessed in either the Novel or Textbook classes.

4.

```
public class Tester
{
    private int[] testArray = {3, 4, 5};

    /** @param n an int to be incremented by 1 */
    public void increment (int n)
    { n++; }

    public void firstTestMethod()
    {
        for (int i = 0; i < testArray.length; i++)
        {
            increment(testArray[i]);
            System.out.print(testArray[i] + " ");
        }
    }

    public void secondTestMethod()
    {
        for (int element : testArray)
        {
            increment(element);
            System.out.print(element + " ");
        }
    }
}
```

What output will be produced by invoking firstTestMethod for a Tester object?

(A) 3 4 5
(B) 4 5 6
(C) 5 6 7
(D) 0 0 0
(E) No output will be produced. An ArrayIndexOutOfBoundsException will be thrown.

5.

What output will be produced by invoking `secondTestMethod` for a `Tester` object, assuming that `testArray` contains `3,4,5`?

(A) `3 4 5`
(B) `4 5 6`
(C) `5 6 7`
(D) `0 0 0`
(E) No output will be produced. An `ArrayIndexOutOfBoundsException` will be thrown.

6.

Consider the following two classes.

```
public class Bird
{
    public void act()
    {
        System.out.print("fly ");
        makeNoise();
    }

    public void makeNoise()
    {
        System.out.print("chirp ");
    }
}
```

```
public class Dove extends Bird
{
    public void act()
    {
        super.act();
        System.out.print("waddle ");
    }

    public void makeNoise()
    {
        super.makeNoise();
        System.out.print("coo ");
    }
}
```

Suppose the following declaration appears in a class other than `Bird` or `Dove`:

```
Bird pigeon = new Dove();
```

What is printed as a result of the call `pigeon.act()`?

(A) `fly`
(B) `fly chirp`
(C) `fly chirp waddle`
(D) `fly chirp waddle coo`
(E) `fly chirp coo waddle`

```
public class Point
{
    private int xCoord;
    private int yCoord;

    //constructor
    public Point(int x, int y)
    {
        ...
    }

    //accessors

    public int get_x()
    {
        ...
    }

    public int get_y()
    {
        ...
    }

    //Other methods are not shown.

}

public abstract class Quadrilateral
{
    private String labels;    //e.g., "ABCD"

    //constructor
    public Quadrilateral(String quadLabels)
    { labels = quadLabels; }

    public String getLabels()
    { return labels; }

    public abstract int perimeter();
    public abstract int area();
}

public class Rectangle extends Quadrilateral
{
    private Point topLeft;   //coords of top left corner
    private Point botRight;  //coords of bottom right corner

    //constructor
    public Rectangle(String theLabels, Point theTopLeft, Point theBotRight)
    { /* implementation code */ }

    public int perimeter()
    { /* implementation not shown */ }

    public int area()
    { /* implementation not shown */ }

    //Other methods are not shown.
}
```

7

Which statement about the `Quadrilateral` class is *false*?

(A) The `perimeter` and `area` methods are abstract because there's no suitable default code for them.
(B) The `getLabels` method is not abstract because any subclasses of `Quadrilateral` will have the same code for this method.
(C) If the `Quadrilateral` class is used in a program, it *must* be used as a superclass for at least one other class.
(D) No instances of a `Quadrilateral` object can be created in a program.
(E) Any subclasses of the `Quadrilateral` class *must* provide implementation code for the `perimeter` and `area` methods.

8

Which represents correct `/*` ***implementation code*** `*/` for the `Rectangle` constructor?

I
```
super(theLabels);
```

II
```
super(theLabels, theTopLeft, theBotRight);
```

III
```
super(theLabels);
topLeft = theTopLeft;
botRight = theBotRight;
```

(A) I only
(B) II only
(C) III only
(D) I and II only
(E) II and III only

9.

Refer to the `Parallelogram` and `Square` classes below.

```
public class Parallelogram extends Quadrilateral
{
    //Private instance variables and constructor are not shown.
        ...

    public int perimeter()
    { /* implementation not shown */ }

    public int area()
    { /* implementation not shown */ }
}

public class Square extends Rectangle
{
    //Private instance variables and constructor are not shown.
        ...

    public int perimeter()
    { /* implementation not shown */ }

    public int area()
    { /* implementation not shown */ }
}
```

Consider an `ArrayList<Quadrilateral> quadList` whose elements are of type `Rectangle`, `Parallelogram`, or `Square`.
Refer to the following method, `writeAreas`:

```
/** Precondition:  quadList contains Rectangle, Parallelogram, or
 *                 Square objects in an unspecified order.
 *  @param quadList the list of quadrilaterals
 */
public static void writeAreas(List<Quadrilateral> quadList)
{
    for (Quadrilateral quad : quadList)
        System.out.println("Area of " + quad.getLabels()
            + " is " + quad.area());
}
```

What is the effect of executing this method?

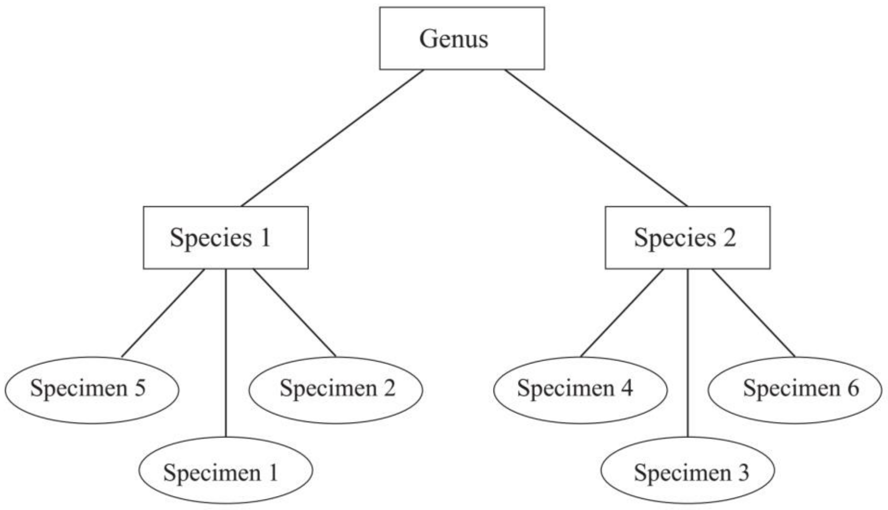
(A) The area of each `Quadrilateral` in `quadList` will be printed.
(B) A compile-time error will occur, stating that there is no `area` method in abstract class `Quadrilateral`.
(C) A compile-time error will occur, stating that there is no `getLabels` method in classes `Rectangle`, `Parallelogram`, or `Square`.
(D) A `NullPointerException` will be thrown.
(E) A `ClassCastException` will be thrown.

Part 2: Free-response questions. (20 marks)

Please answer all the questions in the spaces provided.

A large zoo has a collection of many individual animals of many different species. A computer program is being developed to keep track of all of the animals in the collection.

Because there are so many different kinds of species in the collection, and each species has some unique characteristics and some characteristics in common with other species, it was decided that the computer program would contain objects that correspond to different levels of the taxonomy used by biologists to classify all life forms. A genus is composed of a group of species that have similar common characteristics, as shown in the diagram.



A separate object, `Specimen`, is used to represent each individual animal in the zoo.

The following code implements the Species and Specimen objects:

```
public class Species extends Genus
{
     private String speciesName;
     public Species( String s, String g )
     {
          super(g);
          setSpeciesName(s);
     }

     public void setSpeciesName(String s){ speciesName = s; }
     public String getSpeciesName(){ return speciesName; }
     public String toString()
     {
          return "Species: " + getGenusName() + " " + speciesName;
     }

     public boolean equals(Species s)
     {
          return speciesName.equals(s.getSpeciesName());
     }
}

public class Specimen
{
     private String name;
     private int cageNumber;
     private Species toa; // "Type Of Animal"
     public Specimen( String a, int c, Species s)
     {
          setName(a);
          setCage(c);
          setTOA(s);
     }
     public void setName(String a){ name = a; }
     public void setCage(int c){ cageNumber = c; }
     public void setTOA(Species s){ toa = s; }
     public String getName(){ return name; }
     public int getCage(){ return cageNumber; }
     public Species getTOA(){ return toa; }
     public String toString()
     {
          return name + " is a " + toa + " in cage " + cageNumber;
     }
}
```

10.

(a) State the relationship between the Genus and Species objects. *[1]*

(b) State the relationship between the Species and Specimen objects. *[1]*

(c) Construct the unified modelling language (UML) diagram for the Species object. *[4]*

(d) Outline **two** ways in which the programming team can benefit from the way the relationships between the three objects, Specimen, Species and Genus, have been represented in the code. *[4]*

(e) The Genus class implements a toString() method that produces an output string that is different from the one produced by the toString() method in the Species class.

Consider the following code fragment:

```
Species human = new Species ( "homo", "sapiens" );
System.out.println( human.toString() );
```

(i) Outline why calling the toString() method in this code does not cause an error. *[2]*

(ii) Identify the term for this property. *[1]*

11.

(a) Define the term *encapsulation*. *[1]*

(b) Outline **two** benefits provided by encapsulation. *[4]*

(c) Identify an accessor method in the `Specimen` class. *[1]*

(d) Identify an instance variable in the `Specimen` class. *[1]*

(e) Construct code for the `Genus` object including a constructor, accessor methods and a `toString()` method. *[3]*

The `Specimen` object could have been designed as a sub-class of the `Species` object.

(f) Outline **one** advantage and **one** disadvantage of having the `Specimen` object as a sub-class of the `Species` object. *[4]*

Please answer the questions (a) to (e) in question 10 in the following spaces.

Please answer question 11 in the following spaces.

Part 3: Please submit your written paper and start the coding question on wiki homepage.
Submit your code and screenshot of successful running to your project with a link named:
2016Nov28_Major3_ProgrammingTest_YourName_