Examination Cover Page

Examination Period:	2018 HE Term 1
Academic Institution:	Central Queensland University
Academic Group:	Higher Education Division
Academic Career:	Undergraduate
Examination Type:	2018 HE Term 1 Standard

Affix Student ID Sticker here

I have read and understood the penalties involved if I do not abide by the rules outlined on the back of this examination paper.

Student Signature: _____   Student ID Number: ☐☐☐☐☐☐☐☐☐

Unit:	Object Oriented Programming
Subject Area:	COIT
Catalog Number:	11134
Paper Number:	1
Component:	ALL Components

Duration	180 minutes	Exam Conditions	Open Book
Perusal Time	15 minutes		
First Contact	Umapathy Venugopal	Contact Number	(02) 9324 5789
Second Contact	Mary Tom	Contact Number	(07) 3295 1119

Office Use: Release examination paper via the CQ University Past Exams website two weeks after the DE/SE examination period? Yes

---

Instructor Authorised/Allowed Materials

Calculator - non-programmable, no text retrieval, silent only
Dictionary - non-electronic, concise, direct translation only (dictionary must not contain any notes or comments).

Student Calculator - Make:_____ / Model: _____

Special Instructions to Students:

Please see instruction sheet on first page of the examination paper.

Examination Office Supplied Materials

1 x Rough Paper
2 x Exam Answer Booklet

Questions Answered	Marks	Questions Answered	Marks

Number of examination answer booklets used:
Number of separate sheets attached (Do not include rough paper):

Term 1 Standard Examination 2018

Object Oriented Programming — COIT11134

Instructions Sheet

1.   Write all answers in the Examination Answer Booklet provided.

2.   This examination comprises two parts - Part A and Part B.

3.   Part A is worth 20 marks. Students must answer all questions in this part.

4.   Part B is worth 45 marks. Students must answer all questions in this part.

5.   Write your answers clearly. Use numbered headings or subheadings to show which part of your answer refers to which question. Example: Part A – Question 2.

Term 1 Standard Examination 2018

Object Oriented Programming — COIT11134

PART A   **20 MARKS**

COMPULSORY QUESTIONS
Answer ALL questions from this part.
Each question is worth 2.0 marks (10 x 2 = 20marks).

Briefly comment on the following statements from Java language point of view.
For each question, limit your answers to half a page only. If your answer is
'yes' or 'no', then provide appropriate justification.

Question 1   **2 Marks**

Differentiate instance variables from static variables.

Question 2   **2 Marks**

List the different access modifiers used in Java (**Note**: You need not explain

them).

Question 3   **2 Marks**

A Java class cannot have more than one constructor in it.

Question 4   **2 Marks**

Only one catch block should be written for a try block.

Question 5   **2 Marks**

A Java inner class cannot access the instance fields in the outer (containing) class.

Question 6   **2 Marks**

If there is no constructor specified in a class, objects of that class cannot be
created.

Question 7   **2 Marks**

A sub class can inherit static methods of its super class.

Question 8   **2 Marks**

In Java, a sub class can extend many super classes.

Question 9   **2 Marks**

In Java, a class can implement only one interface.

Question 10   **2 Marks**

Java abstract class can contain implemented and abstract methods.

Object Oriented Programming — COIT11134

PART B   **45 MARKS**

Answer all questions from this part.

Question 1   **5 Marks**

The following SoilType class has incomplete codes. Complete the missing lines of codes for the given parameterized constructor, isOptimumForGardening () method and toString() method.

Note: Soil acidity is measured in pH value on a scale of 1 to 14. If the soil pH is in the range of 5.5 to 7.0 then it is considered to be optimum for gardening. The overridden toString() method should return a string containing the soil pH value and whether it is suitable/optimum for gardening or not.

```java
public class SoilType
{
    private double soilPH;
    private static final double UPPER_LIMIT = 7.0;
    private static final double LOWER_LIMIT = 5.5;

    //constructors
    public SoilType() {
    }
    public SoilType(double soilPH) {
        //Missing lines of code have to be completed by the student



    }
    //getters and setters
    public double getSoilPH() {
        return soilPH;
    }
    public void setSoilPH(double soilPH) {
        this.soilPH = soilPH;
    }
    // Ascertain the PH is in the range of 5.5 to 7.0 so that the soil is suitable for gardening
    public boolean isOptimumForGardening(){
        //Missing lines of code have to be completed by the student



    }
    // Overridden toString Method
    @Override
    public String toString() {
        //Missing lines of code have to be completed by the student



    }
}//end class definition
```
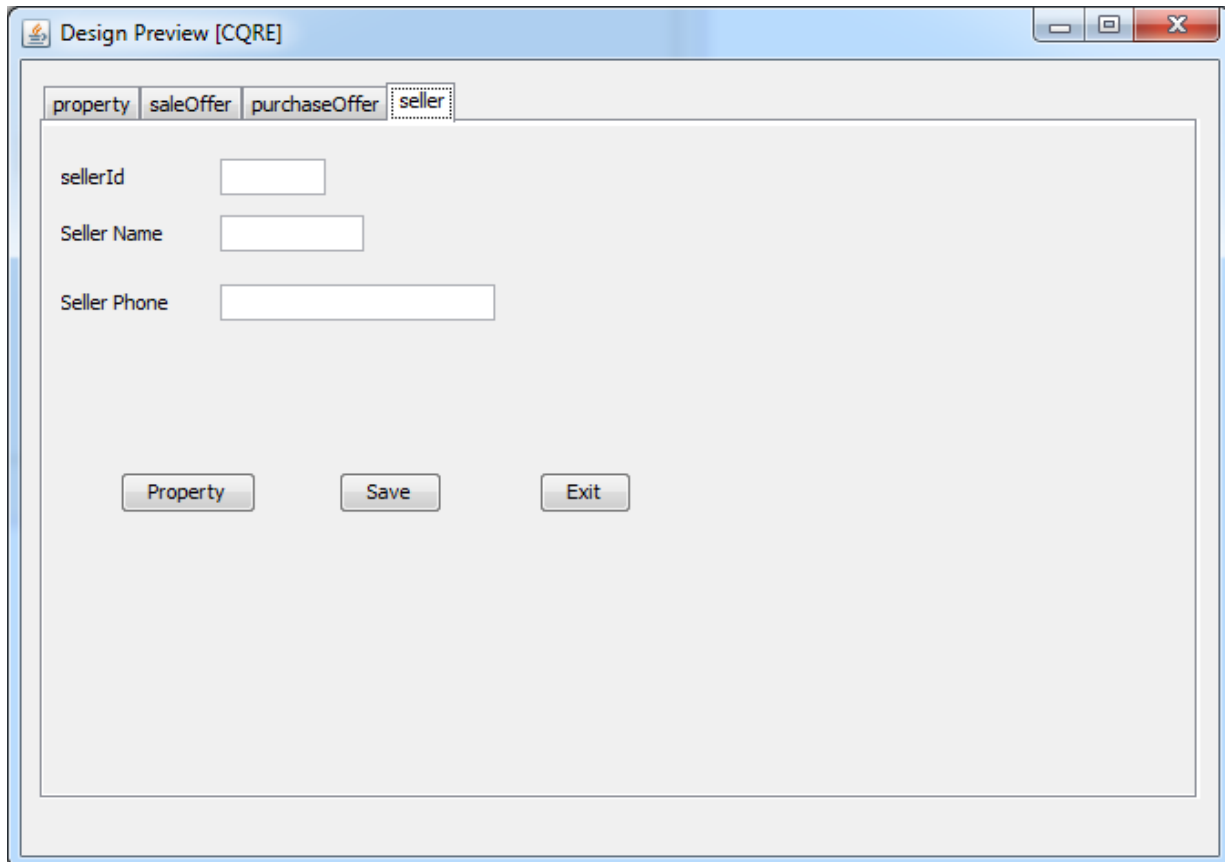
Question 2   **5 Marks**

List all the javax.swing components that have been used to create the
following graphical user interface(GUI).

Question 3   **7 Marks**

The following table provides the characteristics of the fields that are used in the House and Building classes.

Field name	Class	Access specifier	Data type
address	Building	private	String
buildingValue	Building	private	double
numberOfBedRooms	House	private	double
numberOfToilets	House	private	double

You need to write (define) Building class as an abstract class and House class as a derived/sub class of Building. Your classes should have appropriate fields/variables, constructors, getters and setters methods and overridden toString() methods.

Question 4   **3 Marks**

Write a test driver to create two house objects by using the House class that has been mentioned in the above question 3. The test driver should also contain suitable codes to print the details of those house objects appropriately on the screen.

Question 5   **5 Marks**

Complete the missing method implementation in the following Company class so that the following program produces the output shown on execution.

ABC Limited needs to pay $42000.00 tax

```
//Company class is implementing TaxPayable interface
public class Company implements TaxPayable{

    // fields or variables
    private String name;
    private double income;
    private double expenses;

    //parameterized constructor
    public Company (String name, double income, double expenses) {
      this.name = name;
      this.income = income;
      this.expenses = expenses;
    }

    // getter to return the company name
    public String getName() {
       return name;
    }
```

/* Implement the method findTax() of TaxPayable interface to return tax amount that is calculated by the following formulae:
Tax amount = (income – expenses) X TAX_RATE
***/**

//Missing method has to be completed by the student

```
┌─────────────────────────────────────────────────────────────────┐
│                                                                   │
│                                                                   │
│                                                                   │
└─────────────────────────────────────────────────────────────────┘
```

```
    // creating and testing a company object
    public static void main (String [] args) {
       Company account1 = new Company ("ABC Limited", 240000, 100000);
       System.out.printf("%s needs to pay $%.2f tax\n", account1.getName(), account1.findTax());
    } // end of main method
} // end of company class

//TaxPayable interface has an abstract method and a constant for company tax rate
interface TaxPayable{
    double TAX_RATE = 0.3;
    double findTax();

}// end of TaxPayable interface
```

Question 6   **5 Marks**

List the restrictions and limitations that have to be considered when working with type variable 'T' and wild card '?' in Java generics.

Question 7   **5 Marks**

The following Java program uses the data structure **Stack** to store data and the program runs without any error. Write the output produced by the program run.

```java
import java.util.Stack;
public class StackTest
{
    public static void main(String []args)
    {
         // name of OSI layers in an array
         String [] OSI = {"Application", "Presentation", "Session", "Transport", "Network",
                    "DataLink", "Physical"};

        //creating an object of Stack namely inPacket and place the contents of above array in it
        Stack<String>  inPacket  =new Stack<String>();

        for(int i=0; i< OSI.length; i++)
           inPacket.push(OSI [i]);

        /* creating outPacket, taking the contents from the inPacket, one by one, and place them in
        outPacket */

        Stack <String> outPacket =new Stack <String>();
        while(inPacket.size()>0)
           outPacket.push(inPacket.pop());

        //printing the name of layers from inPacket and outPacket
        System.out.println("inPacket contains ");
        while(inPacket.size()>0)
            System.out.println(inPacket.pop());

        System.out.println("outPacket contains ");
        while(outPacket.size()>0)
            System.out.println(outPacket.pop());
    } // end of main method
} // end of class
```

Question 8 **5 Marks**

Identify and correct any **five (5) errors** in the following class Pet. You need not rewrite the entire program again. In your answer, mention each of the line numbers that contains the error and appropriate corrected code.

Hint: The expected output by this (corrected error free) program is as follows:

Total number of Pets = 4
Total number of Dogs = 1
Total number of Cats = 2

//Pet class has to display the number of pets and number of dogs and number of cats.

```
1.  public class Pet {
2.      private String petName;
3.      private String petType;
4.      private static int numPets = 0;
5.      private static int numDogs = 0;
6.      private static int numCats = 0;
7.      //constructor
8.      public Pet(String petName, String petType) {
9.          this.petName = petName;
10.         this.petType = petType;
11.         checkPetType();
12.     }
13.     //checking the pet type and incrementing pets, dogs and cats appropriately
14.     public void checkPetType() {
15.         numPets++;
16.         if(petType.equalsIgnoreCase("Dog"))
17.             numDogs++;
18.         else if (petType.equalsIgnoreCase("Cat"))
19.             numCats++;
20.     }
21.     //getters of number of pets, number of cats and number of dogs
22.     public static void getNumPets() {
23.         return numPets;
24.     }
25.     public static void getNumDogs() {
26.         return numDogs;
27.     }
28.     public static void getNumCats() {
29.         return numCats;
30.     }
```

Question 8 (continued)

```
31.    //test driver
32.    public static void main( String [] args) {
33.            // creating Pet objects
34.            Pet a1 = new Pet( "King", "cobra");
35.            Pet a2 = new Cat("Majestic", "cat");
36.            Pet a3 = new Pet("Appu", "Dog");
37.            Pet a4 = new Cat("Felix", "Cat");
38.            // printing number of pets
39.            System.out.println("Total number of Pets = %d\n",Pet.getNumPets());
40.            // printing number of dogs
41.            System.out.printf("Total number of Dogs = % d\n", Pet.getNumDogs());
42.            // printing number of cats
43.            System.out.printf("Total number of Cats = % d\n", Pet.getNumCats());
44.    } // end of main
45. }// end of Pet class
```

Question 9   **5 Marks**

The ShapeTest class contains incomplete codes in the main method. You need to complete those missing lines of codes to test the polymorphic behaviour by calling the calculateArea method appropriately.

```
interface Shape
{
        public double calculateArea();
}

class Square implements Shape {
   // field/variable
   double length;

   //constructor
   public Square(double length)
   {
      this.length = length;
   }

   // returning the area of square
   public double calculateArea(){
      return length * length;
   }
}// end of Square class
```

Question 9 (continued)

```java
//Circle class
class Circle implements Shape {

    // field/variable
    double radius;

    //constructor
    public Circle(double radius)
    {
        this.radius = radius;
    }


    // returning area of circle

    public double calculateArea(){
       return Math.PI * radius * radius;
    }
}// end of Circle class

// Test driver to test the objects of Square and Circle
public class ShapeTest {
  public static void main (String [] args){

    /* Create an array to hold two (2) shape (implementing) objects and assign its first
    element to a square object and the second element to a circle object.  Test the
    polymorphic behaviour by calling calculateArea() method appropriately
    */
```

//Missing lines of codes have to be completed by the student

```java
    } // end of main method

} // end of test driver
```

--End of ExamPaper--