**COMP3021 Java Programming**
**Spring 2016**
**Midterm Exam**
**23/03/2016**
**Time Limit: 80 Minutes**

**Name:** _____
**Stu ID:** _____
**Email Address:** _____

Instructions:

1. This exam contains 16 pages (including this cover page) and 3 questions.
2. This is a closed book exam.
3. Please write only in the exam paper.
4. You can use either pen or pencil.
5. Please take out your student ID card, and leave it at the top left side of your table.

Grade Table (for teacher use only)

| Question | Points | Score |
|---|---|---|
| 1 | 60 | |
| 2 | 8 | |
| 3 | 10 | |
| Total: | 78 | |

**Question 1 [60 points]**

Please choose only **ONE** answer for each of the questions below.

(a) (2 points) When invoking a method with an object argument, which is passed:

A. a copy of the object
B. the object is copied, then the reference of the copied object
C. the reference of the object
D. the contents of the object

(a) **C**

(b) (3 points) Analyze the following code.

```
public class Test {
   int x;

   public Test(String t) {
      System.out.println("Test");
   }

   public static void main(String[] args) {
     Test test = null;
     System.out.println(test.x);
   }
}
```

A. The program has a compile error because x has not been initialized.
B. The program has a compile error because test is not initialized.
C. The program has a runtime NullPointerException because test is null while executing test.x.
D. The program has a compile error because you cannot create an object from the class that defines the object.
E. The program has a compile error because Test does not have a default constructor.

(b) **C**

(c) (3 points) Analyze the following code:

```
public class Test {
   private int t;

   public static void main(String[] args) {
     int x;
     System.out.println(t);
   }
}
```

A. The program compiles and runs fine.
B. The variable t is private and therefore cannot be accessed in the main method.
C. The variable x is not initialized and therefore causes errors.
D. t is non-static and it cannot be referenced in a static context in the main method.
E. The variable t is not initialized and therefore causes errors.

(c) **D**

(d) (3 points) Analyze the following code:

```
public class Test {
   public static void main(String[] args) {
     double radius;
     final double PI= 3.15169;
     double area = radius * radius * PI;
     System.out.println("Area is " + area);
   }
}
```

A. The program compiles and runs fine.
B. The program has no compile errors but will get a runtime error because radius is not initialized.
C. The program has compile errors because the variable radius is not initialized.
D. The program has a compile error because a constant PI is defined inside a method.

(d) **C**

(e) (3 points) Analyze the following code:

```
class Circle {
   private double radius;

   public Circle(double radius) {
     radius = radius;
   }
}
```

A. The program has a compile error because you cannot assign radius to radius.

B. The program will compile, but you cannot create an object of Circle with a specified radius. The object will always have radius 0.

C. The program does not compile because Circle does not have a default constructor.

D. The program has a compilation error because it does not have a main method.

(e) **B**

(f) (2 points) Given the declaration `Circle[] x = new Circle[10]`, which of the following statement is most accurate.

A. x contains an array of ten int values.

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 a reference to an array and each element in the array can hold a reference to a Circle object.

(f) **D**

(g) (2 points) Given the following command

```
java Test "+" 3 "abc" 2
```

How can you get the word `"abc"` in the main method of Test:

A. args[0]

B. args[1]

C. args[2]

D. args[3]

(g) **C**

(h) (4 points) What is the output of the following program?

```
import java.util.Date;

public class Test {
    public static void main(String[] args) {
        Date date = new Date(1234567);
        m1(date);
        System.out.print(date.getTime() + " ");

        m2(date);
        System.out.println(date.getTime());
    }

    public static void m1(Date date) {
        date = new Date(7654321);
    }

    public static void m2(Date date) {
        date.setTime(7654321);
    }
}
```

A. 7654321 1234567
B. 7654321 7654321
C. 1234567 1234567
D. 1234567 7654321

(h) **D**

(i) (2 points) Which of the following is incorrect?

A. A constructor may be static.
B. A constructor may be private.
C. A constructor may invoke a static method.
D. A constructor may invoke an overloaded constructor.
E. A constructor invokes its superclass no-arg constructor by default if a constructor does not invoke an overloaded constructor or its superclass?s constructor.

(i) **A**

(j) (4 points) Analyze the following code:

```
public class Test {
  public static void main(String[] args) {
    String firstName = "John";
    Name name = new Name(firstName, 'F', "Smith");
    firstName = "Peter";
    name.lastName = "Pan";
    System.out.println(name.firstName + " " + name.lastName);
  }
}

class Name {
  String firstName;
  char mi;
  String lastName;

  public Name(String firstName, char mi, String lastName) {
    this.firstName = firstName;
    this.mi = mi;
    this.lastName = lastName;
  }
}
```

A. The program displays Peter Pan.
B. The program displays John Pan.
C. The program displays Peter Smith.
D. The program displays John Smith.

(j) **B**

(k) (2 points) Which of the following is a correct interface?

A. `interface A { void print() { }; }`
B. `abstract interface A { print(); }`
C. `abstract interface A { abstract void print() { };}`
D. `interface A { void print();}`

(k) **D**

(l) (3 points) What is the output of the following code?

```
public class Test {
  public static void main(String[] args) {
    String s1 = new String("Welcome to Java");
    String s2 = s1;

    s1 += "and Welcome to HTML";

    if (s1 == s2)
      System.out.println("s1 and s2 reference to the same String object");
    else
      System.out.println("s1 and s2 reference to different String
          objects");
  }
}
```

A. s1 and s2 reference to the same String object

B. s1 and s2 reference to different String objects

(l) **B**

(m) (3 points) Suppose you create a class Square to be a subclass of GeometricObject. Analyze the following code:

```
class Square extends GeometricObject {
  double length;

  Square(double length) {
    GeometricObject(length);
  }
}
```

A. The program compiles fine, but you cannot create an instance of Square because the constructor does not specify the length of the Square.

B. The program has a compile error because you attempted to invoke the GeometricObject class's constructor illegally.

C. The program compiles fine, but it has a runtime error because of invoking the Square class's constructor illegally.

(m) **B**

(n) (3 points) Given the following classes and their objects:

```
class C1 {};
class C2 extends C1 {};
class C3 extends C1 {};

C2 c2 = new C2();
C3 c3 = new C3();
```

Analyze the following statement:

```
c2 = (C2)((C1)c3);
```

A. c3 is cast into c2 successfully.

B. You will get a runtime error.

C. You will get a compile time error because multiple casting in nested form is not allowed.

(n) **B**

(o) (3 points) Analyze the following code:

```
class Test {
  public static void main(String[] args) {
    try {
      String s = "5.6";
      Integer.parseInt(s); // Cause a NumberFormatException

      int i = 0;
      int y = 2 / i;
    }
    catch (Exception ex) {
      System.out.println("NumberFormatException");
    }
    catch (RuntimeException ex) {
      System.out.println("RuntimeException");
    }
  }
}
```

A. The program displays NumberFormatException.

B. The program displays RuntimeException.

C. The program displays NumberFormatException followed by RuntimeException.

D. The program has a compilation error.

(o) **D**

(p) (4 points) What is displayed on the console when running the following program?

```
class Test {
  public static void main(String[] args) {
    try {
      method();
      System.out.println("After the method call");
    }
    catch (RuntimeException ex) {
      System.out.println("RuntimeException");
    }
    catch (Exception ex) {
      System.out.println("Exception");
    }
  }

  static void method() throws Exception {
    try {
      String s = "5.6";
      Integer.parseInt(s); // Cause a NumberFormatException

      int i = 0;
      int y = 2 / i;
      System.out.println("Welcome to Java");
    }
    catch (NumberFormatException ex) {
      System.out.println("NumberFormatException");
      throw ex;
    }
    catch (RuntimeException ex) {
      System.out.println("RuntimeException");
    }
  }
}
```

A. The program displays NumberFormatException twice.

B. The program displays NumberFormatException followed by After the method call.

C. The program displays NumberFormatException followed by RuntimeException.

D. The program has a compilation error.

(p) **C**

(q) (4 points) What is the output of running class Test?

```
public class Test {
  public static void main(String[] args) {
    new Circle9();
  }
}

public abstract class GeometricObject {
  protected GeometricObject() {
    System.out.print("A");
  }

  protected GeometricObject(String color, boolean filled) {
    System.out.print("B");
  }
}

public class Circle9 extends GeometricObject {
  /** Default constructor */
  public Circle9() {
    this(1.0);
    System.out.print("C");
  }

  /** Construct circle with a specified radius */
  public Circle9(double radius) {
    this(radius, "white", false);
    System.out.print("D");
  }

  /** Construct a circle with specified radius, filled, and color */
  public Circle9(double radius, String color, boolean filled) {
    super(color, filled);
    System.out.print("E");
  }
}
```

A. ABCD
B. BACD
C. CBAE
D. AEDC
E. BEDC

(q) **E**

(r) (2 points) Suppose A is an interface, B is a concrete class with a default constructor that implements A. Which of the following is correct?

A. `A a = new A();`

B. `A a = new B();`

C. `B b = new A();`

D. `B b = new B;`

(r) **B**

(s) (3 points) Analyze the following code:

```
public class Test1 {
  public Object max(Object o1, Object o2) {
    if ((Comparable)o1.compareTo(o2) >= 0) {
      return o1;
    }
    else {
      return o2;
    }
  }
}
```

A. The program has a compile error because Test1 does not have a main method.

B. The program has a compile error because you cannot cast an Object instance o1 into Comparable.

C. The program would compile if `((Comparable)o1.compareTo(o2) >= 0)` is replaced by `(((Comparable)o1).compareTo(o2) >= 0)`.

(s) **C**

(t) (2 points) Which overrides the following method:

```
protected double xMethod(int x) {...};
```

A. `private double xMethod(int x) {...}`

B. `protected int xMethod(double x) {...}`

C. `public double xMethod(double x) {...}`

D. `public double xMethod(int x) {...}`

(t) **D**

(u) (3 points) Analyze the following code.

```
1. public class Test {
2.   public static void main(String[] args) {
3.     Fruit[] fruits = {new Fruit(2), new Fruit(3), new Fruit(1)};
4.     java.util.Arrays.sort(fruits);
5.   }
6. }

class Fruit {
  private double weight;

  public Fruit(double weight) {
    this.weight = weight;
  }
}
```

A. The program has a compile error on Line 3
B. The program has a runtime error on Line 3
C. The program has a compile error on Line 4
D. The program has a runtime error on Line 4

(u) **D**

**Question 2 [8 points]**

Consider the following program:

```
public class HandleEvent extends JFrame {
  public HandleEvent() {
    // Create two buttons
    JButton jbtOK = new JButton("OK");
    JButton jbtCancel = new JButton("Cancel");
    // Create a panel to hold buttons
    JPanel panel = new JPanel();
    panel.add(jbtOK);
    panel.add(jbtCancel);
    add(panel); // Add panel to the frame
    // Register listeners
1.    OKListenerClass listener1 = new OKListenerClass();
2.    CancelListenerClass listener2 = new CancelListenerClass();
3.    jbtOK.addActionListener(listener1);
4.    jbtCancel.addActionListener(listener2);
  }
  public static void main(String[] args) {
    JFrame frame = new HandleEvent();
    frame.setTitle("Handle Event");
    frame.setSize(200, 150);
    frame.setLocation(200, 100);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }
}
class OKListenerClass implements ActionListener {
  public void actionPerformed(ActionEvent e) {
    System.out.println("OK button clicked");
  }
}
class CancelListenerClass implements ActionListener {
  public void actionPerformed(ActionEvent e) {
    System.out.println("Cancel button clicked");
  }
}
```

(a) (5 points) Combine `OKListenerClass` and `CancelListenerClass` into a single one called `MyListenerClass` by using the method `getSource()` from EventClass. You don't need to copy the whole code, just give the new class and update the "register listeners' part (lines 1-4).

**Solution:**

```
class MyListenerClass implements ActionListener {
```

```
  public void actionPerformed(ActionEvent e) {
    if (e.getSource()==jbtOK)
        System.out.println("OK button clicked");
    else if (e.getSource()==jbtCancel)
        System.out.println("Cancel button clicked");
  }
}
    // Register listeners
1.    MyListenerClass listener = new MyListenerClass();
3.    jbtOK.addActionListener(listener);
4.    jbtCancel.addActionListener(listener);
```

(b) (3 points) Get rid of `OKListenerClass` and `CancelListenerClass` by using anonymous classes. Again you don't need to copy the whole code, just change the "register listeners' part (lines 1-4).

**Solution:**

```
    // Register listeners
3.    jbtOK.addActionListener(new ActionListener {
          public void actionPerformed(ActionEvent e) {
            System.out.println("OK button clicked");
          }
       });
4.    jbtCancel.addActionListener(new ActionListener {
          public void actionPerformed(ActionEvent e) {
            System.out.println("Cancel button clicked");
          }
       });
```

**Question 3 [10 points]**

Implement the following Apple and Person classes as well as any other necessary classes and/or interfaces so that you can sort an array of mixed objects of Apple and Person according to their IDs:

```
Apple:                                Person:
-id: int                              -id: int
-origin: String                       -age: int
-Apple(id:int, origin:String)         -Person(id:int, age:int) // constructors
```

To test your classes, write a Test class with a main() method that creates an array called `arr` of two elements: the first is an apple with id 100 and origin "Hong Kong" and the second a person with id 10 and age 20, sort it with `java.util.Arrays.sort(arr)`, and then print out the array in anyway you like. In essence, the important point

of the question is that you should not write your own sorting algorithm, but call `java.util.Arrays.sort()` which requires you to implement `compareTo()` in the `Comparable` interface:

```
class Apple ... { ... }
class Person ... { ... }
class Test {
  public static void main(String[] args) {
  ... // create arr as an array containing two elements
  java.util.Arrays.sort(arr);
  ... // print out arr
  }
}
// Any other classes/interfacese that you need
```

**Solution:**

```
class ID implements Comparable<ID> {
  int id;
  public int compareTo(ID x) {
      if (this.id > x.id)
     return 1;
      else if (this.id < x.id)
     return -1;
      else return 0;
  }
}
class Apple extends ID {
  String origin;
  Apple(int id, String origin) {
    this.id =id;
    this.origin=origin;
  }
}
class Person extends ID {
  int age;
  Person(int id, int age) {
    this.id =id;
    this.age=age;
  }
}
class Test {
  public static void main(String[] args) {
  ID[] arr = {new Apple(100,"Hong Kong"), new Person(10,20)};
  java.util.Arrays.sort(arr);
  System.out.println(arr[0].id);
```

```
  System.out.println(arr[1].id);
  }
}
```