COMP3021 Java Programming
Spring 2017
Midterm Exam
23/03/2017
Time Limit: 80 Minutes

Name: _____
Stu ID: _____
Email Address: _____

Instructions:

1. This exam contains 19 pages (including this cover page) and 6 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	45	
2	4	
3	5	
4	5	
5	5	
6	10	
Total:	74	

Question 1 [45 points]

Unless stated otherwise, choose only **ONE** answer for each of the questions below.

(a) (2 points) Consider

```
class Test {
    static int x=0;
1   Test(int x) { x=x+1; System.out.println(x); }
2   public static void main(String[] args) {
3      System.out.println(new Test(1));
   }
}
```

A. Compiling error from line 1
B. Compiling error from line 3
C. Run time error (exception) from line 3
D. No error and print out 0 followed by an address of a Test object
E. No error and print out 1 followed by an address of a Test object
F. No error and print out 2 followed by an address of a Test object
G. No error and print out 0
H. No error and print out 1
I. No error and print out 2

(a) **F**

(b) (2 points) Consider

```
class Test {
    static int x=0;
1   Test(int x) { this.x=x+1; System.out.println(x); }
2   public static void main(String[] args) {
3       System.out.println(new Test(1));
   }
}
```

A. Compiling error from line 1
B. Compiling error from line 3
C. Run time error (exception) from line 3
D. No error and print out 0 followed by an address of a Test object
E. No error and print out 1 followed by an address of a Test object
F. No error and print out 2 followed by an address of a Test object
G. No error and print out 0
H. No error and print out 1
I. No error and print out 2

(b) **E**

(c) (2 points) Consider

```
class Test {
1    Test(int x) { this.x=x+1; System.out.println(x); }
2    public static void main(String[] args) {
3        System.out.println(new Test(1));
    }
}
```

A. Compiling error from line 1
B. Compiling error from line 3
C. Run time error (exception) from line 3
D. No error and print out 0 followed by an address of a Test object
E. No error and print out 1 followed by an address of a Test object
F. No error and print out 2 followed by an address of a Test object
G. No error and print out 0
H. No error and print out 1
I. No error and print out 2

(c) **A**

(d) (2 points) Consider

```
void foo(int x, Object y) {}
int a=1;
Object o="Test";
foo(a,o);
```

When executing `foo(a,o)`:

A. a copy of a is passed to x, and a copy of o is passed to y
B. a copy of a is passed to x, and the reference of o is passed to y
C. the address of a is passed to x, the reference of o is passed to y
D. a copy of a is passed to x, and a copy of o is first created then its reference is passed to y

(d) **B**

(e) (2 points) Analyze the following code.

```
public class Test {
   Object x="Test";

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

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

A. The program has a compile error because x cannot be initialized to a string.
B. The program has a compile error because test is not initialized.
C. The program has a runtime NullPointerException.
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.

(e) **C**

(f) (2 points) Consider

```
class A {
   int t;
   A() { t=1; }
}
class B extends A {
   B() { t=t+1; }
}
B a=new B();
```

A. Compiling error because t is not accessible in B()
B. Compiling error because t is not initiated when it is used in B()
C. No error and a.t is 1
D. No error and a.t is 2

(f) **D**

(g) (2 points) Consider

```
class A {
   private int t;
   A() { t=1; }
}
class B extends A {
   B() { t=t+1; }
}
B a=new B();
```

A. Compiling error because t is not accessible in B()
B. Compiling error because t is not initiated when it is used in B()
C. No error and a.t is 1
D. No error and a.t is 2

(g) **A**

(h) (2 points) Consider

```
class A {
   protected int t;
   A() { t=1; }
}
class B extends A {
   B() { t=t+1; }
}
B a=new B();
```

A. Compiling error because t is not accessible in B()
B. Compiling error because t is not initiated when it is used in B()
C. No error and a.t is 1
D. No error and a.t is 2

(h) **D**

(i) (8 points) **This question has multiple answers.** Given

```
abstract class X { int a; }
interface I { void foo(); }
class Y extends X implements I {
  Y() { }
  public void foo() {};
}
```

Which of the following statements are correct:

A. `X a;`
B. `X a=new X();`
C. `X a=new I();`
D. `X a=new Y();`
E. `I a;`
F. `I a=new X();`
G. `I a=new I();`
H. `I a=new Y();`
I. `X[] a;`
J. `X[] a=new X[10];`
K. `X[] a=new I[10];`
L. `X[] a=new Y[10];`
M. `I[] a;`
N. `I[] a=new X[10];`
O. `I[] a=new I[10];`
P. `I[] a=new Y[10];`

(i) **A D E H I J L M O P**

(j) (2 points) Consider

```
abstract class A {
  abstract public void foo(); }
class B extends A {
  B() { };
  void foo() {};
}
```

A. The program has a compile error
B. The program compiles ok

(j) **A**

(k) (2 points) Consider

```
public interface A {
  void foo(); }
class B implements A {
  B() { };
  void foo() {};
}
```

A. The program has a compile error
B. The program compiles ok

(k) **A**

(l) (2 points) Consider

```
public interface A {
  void foo(); }
class B implements A {
  B() { };
}
```

A. The program has a compile error
B. The program compiles ok

(l) **A**

(m) (3 points) **This question has multiple answers.** Consider

```
public interface X { }
class Y implements X { }
public class Test {
  public static void main(String[] args) {
    X a;
    X b;
    a = new Y();
    b = a;
}
```

Which of the following statements are true:

A. (a instanceof X) is true
B. (a instanceof Y) is true
C. (b instanceof X) is true
D. (b instanceof Y) is true

(m) **A B C D**

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

```
public class Test {
  public static void main(String[] args) {
      String[] name1 = {"John", "F", "Smith"};
    Name name2 = new Name(name1[0], "F", "Smith");
    name1[0] = "Peter";
    name2.name[2] = "Pan";
    System.out.println(name2.name[0] + " " + name2.name[2]);
  }
}
class Name {
  String[] name= new String[3];
  public Name(String firstName, String mi, String lastName) {
    this.name[0] = firstName;
    this.name[1] = mi;
    this.name[2] = 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.

(n) **B**

(o) (2 points) Consider

```
public class Test {
  int i,j;
  public void foo() throws Exception {
    try { this.i=this.i/this.j; }
    catch (Exception e) {
      }
  }
  public static void main(String[] args) {
      Test a = new Test();
      a.foo();
  }
}
```

Which of the following statements is true

A. There is a compiling error
B. It compiles ok but have a runtime error
C. It compiles and runs fine.

(o) **A**

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

```
class Test {
  public static void main(String[] args) {
    try {
      foo()
      int i = 0;
      int y = 2 / i; // Cause a RuntimeException
    }
    catch (Exception ex) {
      System.out.println("Exception from foo()'');
    }
    catch (RuntimeException ex) {
      System.out.println("RuntimeException");
    }
  }
  public static void foo() throws Exception { throw new Exception(); }
}
```

Which of the following statements is true

A. There is a compiling error

B. It compiles ok but have a runtime error

C. It compiles and runs fine.

(p) **A**

(q) (3 points) Consider the following code:

```
class A {
    public int a = 0;
    public void fun(){
      System.out.print("A");
   }
}
public class B extends A{
   public int a = 1;
   public void fun(){
       System.out.print("B");
   }
   public static void main(String[] args){
        A classA = new B();
        System.out.print(classA.a);
        System.out.print(((B)classA).a);
        classA.fun();
    }
}
```

What is the output for the code above?

A. 01A
B. 01B
C. 11A
D. 11B

(q) **B**

Question 2 [**4 points**]

Give the output of 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"); }
    finally {
      System.out.println("Finally from main");
    }
  }
  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"); }
    catch (RuntimeException ex) {
      System.out.println("RuntimeException");
      throw ex; }
    finally {
      System.out.println("Finally from method");
    }
  }
}
```

Solution:

```
NumberFormatException
Finally from method
After the method call
Finally from main
```

Question 3 [5 points]

This question is from Lab #4. Give the output of the following program:

```
class A{
    public String show(D obj){
          return ("A and D");
    }
    public String show(A obj){
          return ("A and A");
    }
}
class B extends A{
   public String show(B obj){
          return ("B and B");
    }
    public String show(A obj){
          return ("B and A");
    }
}
class C extends B{}
class D extends B{}
public class midterm {
   public static void main(String[] args) {
      A a = new B(); // Be careful: a is new B()
      B b = new B();
      C c = new C();
      D d = new D();
      System.out.println(a.show(a));
      System.out.println(a.show(b));
      System.out.println(a.show(c));
      System.out.println(a.show(d));
   }
}
```

Solution:

```
B and A
B and A
B and A
A and D
```

Question 4 [5 points]

We have seen that sometimes we can eliminate the use of anonymous inner class in codes by using lambda expressions. For example, we can replace the following code

```
btOpen.setOnAction(new EventHandler<ActionEvent>() {
  public void handle(ActionEvent e) {System.out.println("Process Open");}});
```

by

```
btOpen.setOnAction((ActionEvent e) -> {System.out.println("Process Open");});
```

You have seen the following event handling code for controlling circles:

```
public class ControlCircle extends Application {
  private CirclePane circlePane = new CirclePane();
  public void start(Stage primaryStage) {
    // Hold two buttons in an HBox
    HBox hBox = new HBox();
    hBox.setSpacing(10);
    hBox.setAlignment(Pos.CENTER);
    Button btEnlarge = new Button("Enlarge");
    hBox.getChildren().add(btEnlarge);
    // Create and register the handler
*   btEnlarge.setOnAction(new EnlargeHandler());
    BorderPane borderPane = new BorderPane();
    borderPane.setCenter(circlePane);
    borderPane.setBottom(hBox);
    BorderPane.setAlignment(hBox, Pos.CENTER);
    Scene scene = new Scene(borderPane, 200, 150);
    primaryStage.setTitle("ControlCircle"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }
  class EnlargeHandler implements EventHandler<ActionEvent> {
    @Override // Override the handle method
    public void handle(ActionEvent e) {
      circlePane.enlarge();
    }}
}
```

Here we see that in line (*), we create an `EventHandler` object, and register it with the source object `btEnlarge`. Now replace (*) by a lambda expression:

Solution:

```
btEnlarge.setOnAction((ActionEvent e) -> { circlePane.enlarge();});
```

Question 5 [5 points]

Consider the following incomplete code:

```
class DeepCopy {
    int size;
    java.util.Date date;
    DeepCopy(int size, java.util.Date date) {
   this.size=size;
   this.date=date;
    }
    // override the clone() method in Object class
    public Object clone() {
      // your implementation here
    }
    public static void main(String[] args) {
   DeepCopy a = new DeepCopy(2,new java.util.Date());
   DeepCopy b = (DeepCopy) a.clone();
   if (a==b) System.out.println("a==b is true");
       else System.out.println("a==b is false");
   if (a.date==b.date) System.out.println("a.date==b.date is true");
       else System.out.println("a.date==b.date is false");
   if (a.date.equals(b.date)) System.out.println("a.date equals b.date is
      true");
       else System.out.println("a.date equals b.date is false");
    }
}
```

Provide an implementation of clone() to make a deep copy of DeepCopy objects so that when DeepCopy is run, we have the following desired output:

```
a==b is false
a.date==b.date is false
a.date equals b.date is true
```

Solution:

```
  public Object clone() {
    return new DeepCopy(size,(java.util.Date)date.clone());
    }
```

Question 6 [10 points]

You have seen the following UML diagram in Lab #5:

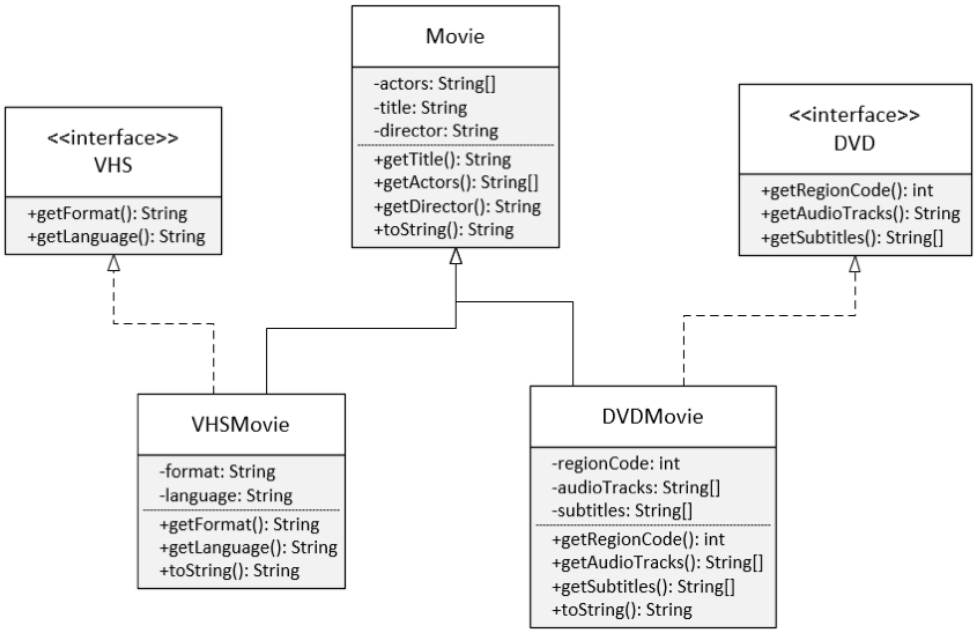

Figure 1: The UML diagram of Video Home System and DVD Movies

For this question, implement the following classes and interfaces:

- Abstract class: Movie
- Interface: VHS
- Class: VHSMovie

Recall that in a UML diagram, "+" means public access and "-" means private access. In addition to the listed methods, don't forget to implement constructors so that

```
new VHSMovie(title, actors, director, format, language);
```

will work. All the getters like `getFormat()` should have the standard imlementation. For VHSMovie, implement `toString()` so that it prints out the title, the format and the language of the movie.

Solution:

```
public abstract class Movie {
 private String title;
 private String[] actors;
 private String director;
```

```
   public Movie(String initialTitle, String[] initialActors, String
       initialDirector){
      this.title = initialTitle;
      this.actors = initialActors;
      this.director = initialDirector;
   }

   public String getTitle(){
      return this.title;
   }
   public String[] getActors(){
      return this.actors;
   }
   public String getDirector(){
      return this.director;
   }
   public String toString(){
      return "title="+this.title;
   }
}
```

```
public interface VHS {
   public String getFormat();
   public String getLanguage();
}
```

```
public class VHSMovie extends Movie implements VHS {
   private String format;
   private String language;
   public VHSMovie(String title, String[] starring,
         String director, String format2,
         String language2) {
      // TODO Auto-generated constructor stub
      super(title, starring, director);
      this.format = format2;
      this.language = language2;
   }
   public String getFormat(){
      return format;
   }
   public String getLanguage(){
      return language;
   }
   @Override
   public String toString() {
```

```
        // TODO Auto-generated method stub
        return super.getTitle()+","+this.format+","+this.language;
    }
}
```