Ryerson University
Department of Electrical & Computer Engineering
COE318

Final Examination Dec. 3, 2013

Name:_____ Section:_____

Student Number: _____

Time: 3 hours
Circle the name of your Professor: E. Bagheri, O. Das, T. Yang.

Instructions

This exam is 3 hours long. It contains 10 questions. Please check your copy to make sure that it contains 18 pages (including this page) before you start.

(1) This is a closed-book exam.
(2) If in doubt on any question, then you must clearly state your own assumption(s).
(3) For each question, you must show your steps and reasoning clearly to be awarded part marks for your answer.

For marking use only (do not fill)

Questions	Score	Max
1		5
2		5
3		10
4		10
5		10
6		10
7		15
8		15
9		20
10		20
Total		120

Question 1. (5 marks)

What is the output of the following program if you consider `IntObj` to be a class containing one private field, an integer. `int getInt()` returns the value of this integer; `void setInt(int newValue)` changes its value to `newValue`.

```
IntObj p;
p = new IntObj();
IntObj q = new IntObj();
q.setInt(20);
p.setInt(q.getInt());
System.out.println("the value of p is "+p.getInt());
p.setInt(15);
System.out.println(q.getInt());
q = p;
System.out.println(q.getInt());
p = null;
System.out.println(p.getInt());
```

Answer:

the value of p is 20
20
15
Exception in thread "main"
java.lang.NullPointerException

Question 2. (5 marks)

For each of the following questions, circle T (true) or F (false):

T / F You can write a constructor for an abstract class.

T / F You can declare a constructor in an interface.

T / F You can create an instance of an abstract class.

T / F A class can extend only one other class.

T / F A class can implement at most one interface.

Answer:
T
F
F
T
F

Question 3. (10 marks)

Consider the following Java code.

```
 1  public class Test {
 2     public void myMethod() {
 3       try {
 4         _________________;
 5       }
 6       catch( NullPointerException npex ) {
 7         System.out.println("1");
 8       }
 9       catch( IllegalArgumentException iaex ) {
10         System.out.println("2");
11       }
12       System.out.println( "3" );
13     }
14
15     public void fragile(int i) {
16       if(i <= 0)
17         throw new IllegalArgumentException();
18       else if(i == 1)
19         throw new NullPointerException();
20       else if(i == 2)
21         throw new ArithmeticException();
22     }
23
24     public static void main(String[] args) {
25       try {
26         Test t = new Test();
27         t.myMethod();
28       }
29       catch( NullPointerException ex ) {
30         System.out.println( "4" );
31       }
32       finally {
33         System.out.println( "5" );
34       }
35       System.out.println( "6" );
36     }
37  }
```

Indicate the output of this code for the following cases:

(a) If the Line 4 is `fragile(0)` .

Answer:

```
2
3
5
6
```

(b) If the Line 4 is `fragile(1)` .

Answer:

```
1
3
5
6
```

(c) If the Line 4 is `fragile(2)` .

Answer:

```
5
Exception in thread "main"
java.lang.ArithmeticException
        at Test.fragile(Test.java:21)
        at Test.myMethod(Test.java:4)
        at Test.main(Test.java:27)
```

Question 4. (10 marks)

According to the classes below, which of the statements below are legal (that is, they do not generate either runtime or compile time errors) and what is the output of each statement, if any.

```
public class Foo extends Bar {
  public void b() {
    System.out.println("Foo b");
  }
}
public class Bar {
  public void a() {
    System.out.println("Bar a");
  }
  public void b() {
    System.out.println("Bar b");
  }
  public String toString() {
    return "Bar";
  }
}
public class Some extends Yam {
  public void a() {
    System.out.println("Some a");
  }
  public void d() {
    System.out.println("Somed");
  }
}
public class Yam extends Foo {
  public void a() {
    System.out.println("Yam a");
  }
  public String toString() {
    return "Yam";
  }
}
```

Answer:
(a) illegal
(b) legal
(c) legal
legal, Yam a
(d) legal
illegal
(e) legal
legal, Bar a

a) `Yam a = new Foo();`

- ☐ illegal
- ☐ legal

if you selected legal, what is the output (if any): _____

b) `Foo b = new Some();`

- ☐ illegal
- ☐ legal

  if you selected legal, what is the output (if any): _____

c) `Bar c = new Yam();`

- ☐ illegal
- ☐ legal

  if you selected legal, what is the output (if any): _____

`c.a();`

- ☐ illegal
- ☐ legal

  if you selected legal, what is the output (if any): _____

d) `Yam d = new Some();`

- ☐ illegal
- ☐ legal

  if you selected legal, what is the output (if any): _____

`d.d();`

- ☐ illegal
- ☐ legal

  if you selected legal, what is the output (if any): _____

e) `Foo e = new Foo();`

- ☐ illegal
- ☐ legal

  if you selected legal, what is the output (if any): _____

`e.a();`

- ☐ illegal
- ☐ legal

  if you selected legal, what is the output (if any): _____

Question 5. (10 marks)

Consider the following `Series` class:

```
public class Series {

  public int getNumberAt(int n) {
    if (n < 0)
      throw new IllegalArgumentException("invalid arg");

    if (n == 0) return 0;
    if (n == 1) return 1;

    int prevPrev = 0;
    int prev = 1;
    int result = 0;

    for (int i = 2; i <= n; i++) {
        result = prev + prevPrev;
        prevPrev = prev;
        prev = result;
    }
    return result;
  }
}
```

Write the code for the junit test method `testGetNumberAt`() that tests the `getNumbetAt` method by providing it an argument of 5.

Write a JUnit test method `testInvalidGetNumberAt()` that tests the `getNumbetAt` method with an argument of -1. The test should pass only if an IllegalArgumentException is thrown.

Assume that the `testGetNumberAt`() and `testInvalidGetNumberAt()` methods are in class `ATest`.

```
public class ATest {
    public void testGetNumberAt() {
        // Write the code here
```

```
        Series s = new Series();
        int expected = 5;
        assertEquals(expected, s.getNumberAt(5));
```

```
    }

    public void testInvalidGetNumberAt(){
        // Write the code here
```

```
        boolean gotEx = false;
        try {
        Series s = new Series();
            s.getNumberAt(-1);
        catch(IllegalArgumentException e) {
          gotEx = true;
        }
        assertTrue(gotEx);
```

```
    }
}
```

Question 6. (10 marks)
What is the output when the following is executed?

```
public class Person {
  static int alive=0;
  static int population=0;
  private String name;

  //the constructor
  public Person(String tempName) {
    name = tempName;
    population++;
    alive++;
  }

  public Person becomeParent() {
    population++;
    alive++;
    return new Person(this.name + "Child");
  }

  public void die(Person p) {
    alive--;
    p=null;
  }

  public static void main(String[] args) {
    Person adam = new Person("Adam");
    Person eve = new Person("Eve");
    System.out.println(adam.population + " " + eve.alive);
    Person seth=adam.becomeParent();
    Person cain=eve.becomeParent();
    System.out.println(cain.population + " " + eve.alive);
    adam.die(seth);
    cain.die(eve);
    System.out.println(adam.population + " " + eve.alive);
    System.out.println(seth.name);
    System.out.println(cain.name);
  }
}
```

Answer:

2 2
6 6
6 4
AdamChild
EveChild

Question 7. (15 marks)

Consider the following classes:

```
public class A {
    private int i;
    public A() {
        this(1);
        System.out.println("A0:" + i);
    }
    public A(int i) {
        this.i = i;
        System.out.println("A1:" + i);
    }

    public int getI() {
        return i;
    }
}

public class B extends A {
    public B(int i) {
        super(i);
        System.out.println("B1:" + getI());
    }

    public B() {
        this(3);
        System.out.println("B0:" + getI());
    }

    public static void main(String[] args) {
        new B();
        new A();
        new B(5);
    }
}
```

Assume that two classes given above compile without error. What is the output when the `main` method of class B is run?

Answer:

A1:3
B1:3
B0:3
A1:1
A0:1
A1:5
B1:5

Question 8. (15 marks)

How would you fill in the blank spaces if you knew that the output of the main method was the following?

1
12
11
13
16
10

Please note that each blank space may include one or more words in it.

```
_____interface_____  Inter {
   int number();
}


public _____abstract class_____ Abs {

   public int foo = 12;

   int number() {
      return 5;
   }

   abstract  int ace();

   public String toString() {
      return "" + number();
   }
}


public class Super _____extends Abs_____ implements _____Inter_____ {

   int twice(int x) {
      return 2 * x;
   }

   public int thrice(int x) {
      return 3 * x;
   }
```

```
   int ace() {
      return 1;
   }

   String dub(String s) {
      return s + s;
   }

   public int number() {
      return 11;
   }

   public static void main(String args[]) {
      Super s1 = new Super();
      int j = s1.ace();
      System.out.println(j);
      System.out.println(s1.foo);
      System.out.println(s1);
      Super s2 = new Sub(16);
      int i = s2.ace();
      System.out.println(i);
      System.out.println(s2.foo);
      System.out.println(s2);
   }
}



class Sub extends Super {
   Sub(int bar) {
      foo = bar;
   }

   public int number() {
      return 10;
   }

   public int ace() {
      return 13;
   }

   public int dub(int i) {
      return 2 * i;
   }
}
```

Question 9. (20 marks)

There will be fourteen lines of output if we run the main method of the Student class. Please provide exactly those fourteen lines.

```
class Girl extends Student {
  public Girl(int id, Student left, Student right) {
    super(id, left, right);
  }
  public int score() {
    int i = super.score();
    System.out.println(i);
    return i + 3;
  }
}

class Boy extends Student {
  public Boy(int id, Student left, Student right) {
    super(id, left, right);
  }
  public int score() {
    int i = super.score();
    System.out.println(i);
    return i + 2;
  }
}

public class Student {
  private int value;
  private Student left;
  private Student right;

  public Student(int id, Student left, Student right) {
    this.value = id;
    this.left = left;
    this.right = right;
  }
  public int getValue() {
    return value;
  }
  public int score() {
    int v = value;
    if(left != null) {
      v = v + left.getValue();
    }
    if (right != null) {
      v = v + right.getValue();
    }
    return v;
  }
```

```
  public static void main(String[] args) {
    Student a, b, c, d, e, f, g;
    g = new Girl(1, null, null);
    f = new Boy(2, null, null);
    e = new Boy(3, null, null);
    d = new Boy(4, null, null);
    c = new Girl(5, g, f);
    b = new Boy(6, e, d);
    a = new Girl(7, c, b);

    System.out.println(g.score());
    System.out.println(f.score());
    System.out.println(e.score());
    System.out.println(d.score());
    System.out.println(c.score());
    System.out.println(b.score());
    System.out.println(a.score());
  }
}
```

Answer:

1
4
2
4
3
5
4
6
8
11
13
15
18
21

Question 10. (20 marks)

Consider the task of representing tickets to campus events. Each ticket has a unique number and a price. There is only one type of ticket: advance tickets. Advance tickets purchased before the event cost $30 to $40 depending on the date purchased.

To answer this problem, create the following two classes:

(1) Create a class named Ticket. The only instance variable of the class is a `private` int variable: number. All common operations across the ticket types should appear in this class, and all differing operations should be declared in such a way that every subclass must implement them. No actual objects of type Ticket can be created: each actual ticket will be an object of a subclass type.

Implement the following methods:

- The ticket class has a constructor that take a parameter of int variable: number.
- The ticket class has an abstract method *getPrice*.
- The ticket class has a method *toString.* A call on it returns a description ticket number and price. An example string would be "Number: 17, Price: 50.0".

(2) Create a new class named AdvanceTicket as a subclass of Ticket. It has one instance variable daysInAdvance to represent how many days in advance the ticket was purchased.

Implement the following methods:

- The ticket class has a constructor that take two parameter of int variables: number and daysInAdvance.
- The ticket class has a *getPrice* method that returns the cost of an advance ticket. Advance tickets purchased 10 or more days before the event cost $30, and advance tickets purchased fewer than 10 days before the event cost $40.
- The ticket class has a method *toString.* A call on it returns a description an advance ticket number, price and how many days in advance the ticket was purchased. An example string would be "Number: 54, Price: 30.0 (8 days in advance)". Your advance ticket class should have minimal redundancy with your ticket class, and should take advantage of inheritance as appropriate.

Answer:

```
public abstract class Ticket {
    private int number;

    public Ticket(int number) {
        this.number = number;
    }

    public abstract double getPrice();

    public String toString() {
        return "Number: " + this.number + ", Price: " + this.getPrice();
    }
}

public class AdvanceTicket extends Ticket {
    private int daysInAdvance;

    public AdvanceTicket(int number, int daysInAdvance) {
        super(number);
        this.daysInAdvance = daysInAdvance;
    }

    public double getPrice() {
        if (daysInAdvance >= 10) {
            return 30.00;
        } else {
            return 40.00;
        }
    }

    public String toString() {
        return super.toString() + " (" + this.daysInAdvance + " days in advance)";
    }
}
```