CIS 7124 Object-Oriented Programming

Sample Midterm Exam

1. Briefly answer each of the following questions.

   (a) What is an abstract data type? What is the major difference between an abstract data type and a type in procedural programming languages?

   (b) What is the meaning of the keyword `static` in Java? What are the differences between static and non-static members.

   (c) What is polymorphism? Give an example to illustrate the use of polymorphic variables in Java.

2. Program comprehension

2.1 Find errors (either compile or run-time) in the following Java code snippets if there are any. State the type of the error and state explicitly "no" if no error exists.

```
class A {
      public A(int x){}

      public void m(){
            A a = new A();
      }
}
```

(a)

```
class B{
      public void main(String[] args){
            System.out.println(abs(-2));
      }
      static int abs(int x){
            return x<0 ? -x : x;
      }
}
```

(b)

```
import java.util.*;
class C {
      public Stack m(int x){
            Stack s = new Stack();
            s.push((Integer)x);
            return s;
      }
}
```

(c)

```
interface A {
	void m(){
		System.out.println(“m”);
	}
}
```

(d)

```
class A{
	static int x;
	public A(int x){
		this.x = x;
	}
}
```

(e)

```
class A {}
class B extends A {}
class C extends B {}

class D {
	public static void main(String[] args){
		A a = new A();
		C c = new C();
		B b = c;
	}
}
```

(f)

```
class A {
	public A(){}
	public A(A a){}
}
```

(g)

2.2 What are the outputs to the console (standard output) from each of the following programs?

```
class Test {
      int var;
      Test(double var){
            this.var = (int)var;
      }
      Test(int var){
            this(“aaa”);
      }
      Test(String s){
            this();
            System.out.println(s);
      }
      Test(){
            System.out.println(“bbb”);
      }
      public static void main(String[] args){
            Test t = new Test(5);
      }
}
```

(a)

```
class Test {
      int x;
      public Test(int x){
            this.x = x;
      }

      public static void m(Test o){
            o = new Test(2);
      }

      public static void main(String[] args){
            Test o = new Test(1);
            m(o);
            System.out.println(“output=”+o.x);
      }
}
```

(b)

```
class Test {
      public static void main(String[] args){
            System.out.println(“output=”+m((int)1.5));
      }
      static int m(int x){
            return ++x;
      }
}
```

(c)

```
class Test {
      private static int x;
      public Test(){
            x++;
      }

      public static void main(String[] args){
            Test o1 = new Test();
            Test o2 = new Test();
            System.out.println(“output=”+x);
      }
}
```

(d)

```
class J extends SuperJ {
    public J(){
                      System.out.println("ConsJ");
    }
    public void m(){
                        System.out.println("J");
    }

    public static void main(String[] args){
                           SuperJ o = new J();
                                  o.m();
    }
}

class SuperJ {
    public SuperJ(){
                    System.out.println("ConsSuperJ");
    }

    public void m(){
                      System.out.println("SuperJ");
    }
}
```

(e)

3. Programming

3.1 Designing an electronic voting machine is a challenging task. You are asked to implement a very simplified voting machine class with the following specification. A voting machine has a list of candidates and the following methods:

a) `addCandidate(String name)`
/* Add a candidate with the name to the list */

b) `castVote(String name)`
/* Cast a vote to the candidate with the name */

c) `printResults()`
/* Print out the number of votes each candidate has received. The order does not matter */

Assume that only two attributes of a candidate, namely, the name and number of votes, are of interest here, and the size of the candidate list is unknown in advance.

3.2 Implement the following functions (static methods) on linked lists of Integer objects :

a) `public static boolean sorted(LinkedList<Integer> lst)`
/* Return true if the list is sorted in ascending order; otherwise, false. */

b) `public static int sum(LinkedList<Integer> lst)`
/* Return the sum of the elements in the list. */

c)
```
public static LinkedList<Integer>
   replace(LinkedList<Ingeger> lst, Integer o, Integer n)
```
/* Return a copy of the given list, replacing all occurrences of `o` with `n`. Use equals to test equality*/