0. Name (2 pts):

CS 2334: Programming Structures and Abstractions

Final Exam

Thursday, December 17, 2009

General instructions:

- This examination booklet has 15 pages.
- Do not forget to write your name at the top of the page and to sign your name below.
- The exam is open book and notes, but closed electronic device.
- The exam is worth a total of 200 points (and 20% of your final grade).
- Explain your answers clearly and concisely. Do not write long essays (even if there is a lot of open space on the page). A question worth 5 points is only worth an answer that is at most one sentence.
- You have 120 minutes to complete the exam. Be a smart test taker: if you get stuck on one problem go on to the next. Don't waste your time giving details that the question does not request. Points will be taken off for answers containing excessive or extraneous information.
- Show your work. Partial credit is possible, but only if you show intermediate steps.

Problem	Topic	Max	Grade
0	Name	2	
1	Recursion	35	
2	Ethics	40	
3	Exceptions and Assertions	30	
4	Inheritance	25	
5	Generic Programming and Generics	20	
6	Graphical User Interfaces	20	
7	Event-Driven Programming and MVC	10	
8	Collections Framework	20	
Total			

On my honor, I affirm that I have neither given nor received inappropriate aid in the completion of this exam.

Signature: ____________________

Date: ____________________

1. **Recursion** (35 pts)

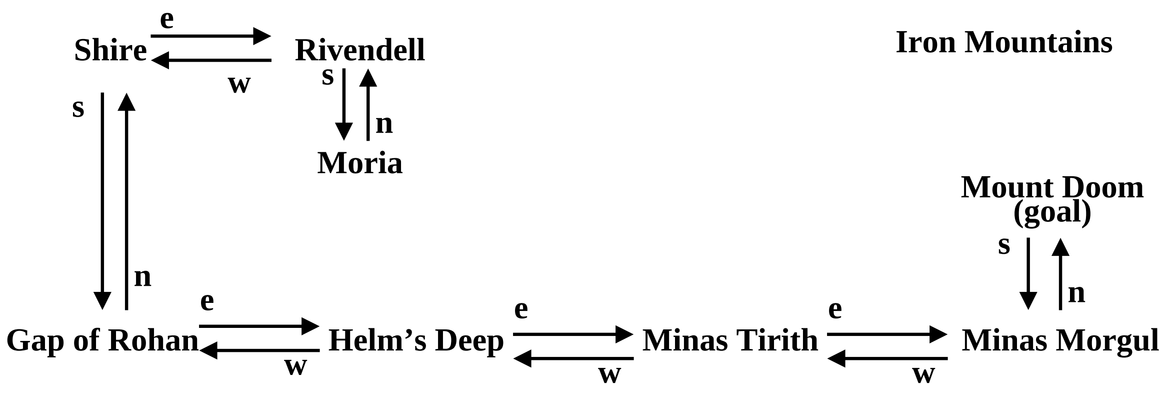
Consider the following class definition:

```
public class Place {
    public boolean goal;     // Is this place a goal?
    public Place north;      // Place to the North (null -> no place)
    public Place south;
    public Place east;
    public Place west;



};
```

We will use this class to represent places on a map. For example, consider the following map:



Here, each of the arrows represent references from one place to another (with n,s,e,w corresponding to North, South, East and West). When no arrow is given for a particular direction, assume that the reference is null.

We will design a method:

```
public boolean isConnected(Place p);
```

that will return **true** if the specified place is somehow connected to a goal place, and **false** otherwise. For example, isConnected(Shire) would return **true** (because Mount Doom is a goal), but isConnected(Iron Mountains) would return **false**. As you answer the following questions, state any other assumptions that you must make.

(a) (5 pts) List the base case(s).

(b) (5 pts) List the recursive case(s).

(c) (5 pts) List the preconditions for isConnected().

(d) (5 pts) List the postconditions for isConnected().

(e) (15 pts) Give the pseudocode for the method.

```
public boolean isConnected(Place p)
{
}
```

2. **Ethics in Computer Science** (40 pts)

(a) (10 pts) True or False and briefly explain: if privacy is a fundamental human right, an individual has the right to sell his/her personal data (thus waiving the right to further control these data).

(b) (15 pts) Researchers at the recent International Conference on Data Mining (ICDM'09) presented work that will allow stores, using video cameras, to track the movements of single customers. Specifically, the marketing team will have access to a time-indexed path (a trajectory) that the customer took during the visit – from entering the store to the checkout line. By seeing where customers tend to go and how long they spend at certain store displays, the team will be able to better plan the layout of the store and the design of displays, so as to improve sales.

What are the key issues and ethical principles that should be considered in the design and deployment of this system?

(c) (15 pts) Briefly explain why the following claim is “muddled” from an ethical and legal perspective: software is property.

3. Exceptions and Assertions (30 pts)

Consider the following code:

```
public class exceptionTest
{
    public void methodA() { ...   // Do some stuff
    };

    public void methodB() { ...   // Do some other stuff
    };

    public void bar() {
        try {
            System.out.println("Foo 5");
            methodB();
            System.out.println("Foo 6");
        }catch(ExceptionB e){
            System.out.println("Foo 7");
            throw e;
        };
        System.out.println("Foo 8");
    }

    static public void main(String[] args) {
        try {
            System.out.println("Foo 1");
            bar();
            methodA();
            System.out.println("Foo 2");
        }catch(ExceptionA e){
            System.out.println("Foo 3");
        };
        System.out.println("Foo 4");
    };
}
```

(a) (10 pts) Suppose that methodB() throws ExceptionB during execution. What output does this program produce and does the program terminate with or without an error? (assume that output is only produced by the println() calls that you see)

(b) (10 pts) Suppose that methodB() throws ExceptionA during execution. What output does this program produce and does the program terminate with or without an error?

(c) (10 pts) Briefly describe the similarities and differences between Exceptions and Assertions.

4. **Inheritance** (25 pts)

Consider the following implementation:

```
public class A
{
    protected String name;

    public A(String name){
        this.name = name;
    }

    public String toString(){
        return("A: " + name);
    };
}

public class B extends A
{
    protected String name;

    public B(String name) {
        super("SUPER-B");
        this.name = name;
    }
}

public class C extends B
{
    private String name;

    public C(String name){
        super("SUPER-C");
        this.name = name;
    };

    public String toString() {
        return("C: " + name);
    };
};

public class driver
{
    public static void main(String args[]) {
    A[] objects = new A[4];

    objects[0] = new A("foo");
    objects[1] = new B("bar");
    C c = new C("baz");
    objects[2] = c;

    for(int i = 0; i < objects.length; ++i) {
        System.out.println(objects[i]);
    };

    B b = c;

    System.out.println(b);
    System.out.println(c);
    };
};
```

(a) (20 pts) What output does this program produce?

(b) (5 pts) Could class A be implemented as an *interface*? Why or why not?

5. **Generic Programming and Generics** (20 pts)

(a) (5 pts) T/F: Generics are enforced at run-time.

(b) (15 pts) Consider the following block of code:

```
public void main(String[] args)
{

   Comparator<B> comp = getComparator();

   A a = new A("A element");
   B b = new B("B element");
   C c = new C("C element");

   comp.compare(a, b);     // Will it compile?

   comp.compare(a, c);     // Will it compile?

   comp.compare(b, c);     // Will it compile?

   comp.equalsTo(b, c);    // Will it compile?
}
```

Assume that getComparator() returns a proper Comparator of the declared type and that this object implements just the methods required by the Comparator interface. Assume that classes A, B, and C are the same as from the previous question. State whether each of the four indicated method calls will compile or not.

6. **Graphical User Interfaces** (20 pts)

(a) (10 pts) What information would the JButton class use in its implementation of the getPreferredSize() method? (we did not explicitly talk about this in class, but you should be able to speculate on some of the details).

(b) (10 pts) True or False and briefly explain: it is possible **and** useful for a single ActionEventListener object to be registered with multiple buttons.

7. **Event-Driven Programming and the Model View Controller Approach** (10 pts)

   (10 pts) Briefly explain the tension between the pure form of the Model View Controller approach and what is often implemented.

8. **Java Collections Framework** (20 pts)

Consider the following code:

```
import java.util.*;

public class collectionTest
{
    public static void main(String[] args)  {
    Set<Integer> col1 = new TreeSet<Integer>();
    col1.add(9);
    col1.add(3);
    col1.add(19);
    col1.add(11);
    col1.add(3);

    List<String> col2 = new LinkedList<String>();
    col2.add("Mirkwood");
    col2.add("Fangorn");
    col2.add("Pinnath Gelin");
    col2.add("Fangorn");
    col2.add("Blue Mountains");

    for(Object element: col1) {
        System.out.println(element.toString());
    };

    Iterator it = col2.listIterator();

    while(it.hasNext()) {
        System.out.println(it.next().toString());
    };
  }
}
```

(20 pts) What output does the program produce?