Final Examination Problem Sheet

TIME: 06/16/2009, 14:20–17:20

This is a closed-book exam. Any form of cheating or lying will not be tolerated. Students can get zero scores and/or fail the class and/or be kicked out of school and/or receive other punishments for those kinds of misconducts. Both English and Traditional Chinese (if suited) are allowed for answering the questions. We do not accept any other languages.

1 POO Zoo

The two unicorn friends (Pinky and Bluey) of Charlie the unicorn are persuading him to come to the POOZoo, the POOCasino, or the POOBoard, instead of the candy mountain. To add those popular characters to POO, one of the members in your programming team wrote the following code:

```java
class Unicorn{
  int x, y;
  static void walk(Unicorn u){ u.x += 2; }
}
class Charlie{
  private int count;
  String reject(String place) { return "I don't want to go to " + place; }
  String agree(String place){ return "Okay, okay, I'll go to " + place; }
  public void do_unicorn_walk(){ Unicorn.walk(this); }
  public String action(String place){
    do_unicorn_walk(); count++;
    if (count < 2) return reject(place);
    else { count = 0; return agree(place); }
  }
}
class Pinky{
  public void do_unicorn_walk(){ Unicorn.walk(this); }
  public String action(String place){
    do_unicorn_walk();
    return "Charlie, Charlie, let's go to " + place;
  }
}
class Bluey{
  public void do_unicorn_walk(){ Unicorn.walk(this); }
  public String action(String place){
    do_unicorn_walk();
    return "Yes, Charlie, " + place;
  }
}
```

The other programmer in your team write the following code:

```java
  class Demo1{
    public static void main(String[] argv){
      Unicorn[] uarr = new Unicorn[3];
      uarr[2] = new Charlie(); uarr[1] = new Bluey(); uarr[0] = new Pinky();
      for(int t = 0; t < 2; t++)
        for(int i = 0; i < uarr.length;i++)
          System.out.println(uarr[i].action("the POO Zoo"));
    }
  }
```

You want the two pieces of code, when coupled together, can show the correct output like

```
Charlie, Charlie, let's go to the POO Zoo
```

```
Yes, Charlie, the POO Zoo
I don't want to go to the POO Zoo
Charlie, Charlie, let's go to the POO Zoo
Yes, Charlie, the POO Zoo
Okay, okay, I'll go to the POO Zoo
```

However, the code above doesn't work, because it cannot pass the compiler (hahaha)!

(1) (5%) Assume that you can ask another programmer to re-write the latter piece of code, while the former remains unchanged. Write down the *correct* and *clean* lines that you expect her/him to write to cope with the former piece.

(2) (15%) Assume that you can ask another programmer to re-write the former piece of code, while the latter remains unchanged. Write down the *correct* and *clean* lines that you expect her/him to write to cope with the latter piece.

Note that the shortness of your answers is a big concern, of course, because you are in a time-bounded exam. The quality of your answers would also be taken into account.

2 POO Calculus

Yes, yes, yes. It is purely evil to remind you that your calculus exam is coming soon. But imagine that you can bring your well-designed POO Calculus code to the JVM of your cellphone to the calculus exam, wouldn't it be fantastic? Let's start developing this useful software for future students.

One basic component in calculus is the real-valued function from $\mathbb{R}$ to $\mathbb{R}$. We will define it as an abstract Java class to begin with.

```java
abstract class Function{ public abstract double eval(double x); }
```

A (symbolically) differentiable function $f(x)$ is a function such that $f'(x)$ has a closed form; a (symbolically) integrable function $g(x)$ is a function such that $\int g(x)dx$ has a closed form. The two properties would be represented by the following interfaces.

```java
interface Differentiable{ Function diff(); }
interface Integrable{ Function inte(); }
```

Thus, a function $h(x) = \exp(-0.5 \cdot x^2)$ would probably be like

```java
class SDExp extends Function implements Differentiable{
  public double eval(double x){ return Math.exp(-0.5 * x * x); }
  public Function diff(){ return new Multiply(new Poly(-1.0, 1), this); }
}
```

(1) (5%) We know that the exponential function $g(x) = \exp(x)$ is both differentiable and integrable, with $g'(x) = g(x) = \int g(x)dx$. Complete the code below.

```java
class Exp extends Function implements Differentiable, Integrable{
  //write your code
}
```

(2) (5%) We know that the simple polynomial function $g(x) = \alpha x^D$ is both differentiable and integrable, with $g'(x) = \alpha D x^{D-1}$ and $\int g(x)dx = \frac{\alpha}{D+1}x^{D+1}$. Complete the code below.

```java
class Poly extends Function implements Differentiable, Integrable{
  double deg;
  double coef;
  Poly(double coef, double degree){ this.coef = coef; this.deg = deg; }
  //write your code
}
```

(3) (5%) The `Sum` class represents the sum of two functions. Use the fact that $(f+g)' = f' + g'$ and $\int(f+g) = \int f + \int g$ to complete the code below. *What would happen if either* `f` *or* `g` *is not symbolically differentiable?*

```
class Sum extends Function implements Differentiable, Integrable{
  Function f, g;
  Sum(Function f, Function g){ this.f = f; this.g = g; }
  //write your code
}
```

(4) (5%) Without simplifying any equations manually, write the code that computes the value of

$$\left.\frac{d\big(3x^2+2x+5+(\int \exp(x)dx)\big)}{dx}\right|_{x=2}.$$

Note: You can use `Math.exp(double a)`*, which returns* $\exp(a)$*. You can also use* `Math.pow(double a, double b)`*, which returns* a^b*.*

3 POO Search

POO is deciding to make many things searchable—boards, articles, friend lists, messages, to name a few. Of course, one wouldn't expect to waste time on writing the same search program several times. Hence, Java Generics is the time-saver (?!). Your programming team decides to start by having a `Searchable` interface, where its `match` function determines whether the instance matches the query string.

```
interface Searchable{ boolean match(String query); }
class POOBoard implements Searchable{ /* ...*/ }
class POOFriend implements Searchable{ /* ...*/ }
class POOGoodFriend extends POOFriend{ /* ...*/ }
```

Now, the team wants a `Searcher` class that works with anything `Searchable`. A naïve idea is

```
class Searcher extends ArrayList<Searchable>{ /* ... */ }
```

However, such a `Searcher` allows us to mix `POOBoard` and `POOFriend` altogether in the list, which does not look very right. What the team really wants is something like

```
class Searcher<E> extends ArrayList<E>{ /* ... */ }
```

But that doesn't look right, either, because we can then have something like

```
Searcher<Integer> s = new Searcher<Integer>();
s.add(new Integer(3));
```

Nevertheless, `Integer` does not implement `Searchable` and hence should not be allowed by `Searcher`.

(1) (10%) Write down the declaration of `Searcher` that would allow it to work on only any `Searchable` type but not a mixture of them. Your declaration should look like

```
class Searcher<E______> extends ArrayList<E______>
```

where the underlines should be replaced by your own code or removed.

(2) (5%) The core method of `Searcher` is a method called `search`, which looks like

```
Searcher<E_______> search(String query){
  Searcher<E_______> result = new Searcher<E_______ >();
  for(int i=0;i<this.size();i++){
    E o = this.get(i);
     if (o.match(query)){ result.add(o); }
  }
  return result;
}
```

Fill in the necessary part in the underlines above that would allow the code below to work.

```
Searcher<POOBoard> s = new Searcher<POOBoard>();
//add some POOBoards to s
Searcher<POOBoard> res = s.search("Gossip");
```

(3) (5%) Another method of `Searcher` is called `addSearchResult`, which looks like

```
Searcher<________> addSearchResult(Searcher<________> orig, String query){
  this.addAll(orig.search(query));
  return this;
}
```

Fill in the necessary part in the underlines above that would allow the code below to work.

```
Searcher<POOGoodFriend> sg = new Searcher<POOGoodFriend>();
//add some POOGoodFriends to sg
Searcher<POOFriend> s2 = new Searcher<POOFriend>();
Searcher<POOFriend> r2 = s2.addSearchResult(sg, "CharlieL");
```

4 POO Exception

Consider the following two exceptions that are used in POO.

```
class NetworkDisconnectException extends IOException{ }
class UserNotExistException extends Exception{ }
```

The following hierarchy in Java might also be helpful.

```
class Exception extends Throwable{ }
class IOException extends Exception{ }
class RuntimeException extends Exception{ }
class NullPointerException extends RuntimeException{ }
```

(1) (20%, with partial credit) A lazy programmer in your team wrote the following exception handling code, which, of course, is not of the top quality. Please use the correct `try-catch-finally` mechanism to make the code better-organized while doing exactly the same thing (if exception only happens during `action_1`, `action_2`, and `action_3`).

```
try{
  action_1();
  if (action_2()){ action_3(); }
  else{ action_0(); return; }
}
catch(Throwable e){
  action_4();
  if (e instanceof NullPointerException){ action_5(); }
  if (e instanceof Exception){ action_6(); }
  if (e instanceof NetworkDisconnectException){ action_7(); }
  if (e instanceof UserNotExistException){ action_8(); }
  action_0();
}
```

Note that the shortness of your answers is a big concern, of course, because you are in a time-bounded exam. The quality of your answers would also be taken into account.

5 POO Love

It is nearly the Lunar Valentine's Day, and POO BBS decides to have an activity of "love sending." Basically, any user is allowed to send love messages to as well as to receive love messages from other users. To handle the process, every user is modeled as an instance of `POOUser` that contains an instance variable `love_count`, which is increased by 1 whenever a love message is received. A `POOLoveThread` is associated with each user to handle the sending process.

```
class POOUser{
  private int love_count;
  public void increase_love_count(){ int r = love_count + 1; love_count = r;}
}

class POOLoveThread extends Thread{
  POOUser user;
  POOLoveThread(POOUser user){ this.user = user; }
  void increase_love_count(POOUser receiver){
    receiver.increase_love_count();
  }
  public void run(){
    while(true){
      POOUser receiver = find_user(ask_receiver(user));
      increase_love_count(receiver);
    }
  }
}
```

In the stories below, user M is a freshman who is competing for the title of "most-loved user" and hence is very serious about his love counts.

(1) (5%) When running the code above, M complains that he should have gotten three love messages from three different junior users $J1$, $J2$, $J3$. Nevertheless, M's `love_count` only shows 1. Is it possible? If so, give an example of three concurrent calls to `M.increase_love_count()` that results in such a behavior. If not, state your arguments to M.

(2) (5%) A programmer in your team tried to respond to M's complaint by adding the keyword `synchronized` to the declaration of the `increase_love_count` method in `POOLoveThread`. One day later, M comes back and has the same complaint again. Is it possible? Why or why not?

(3) (5%) Another programmer in your team tried to respond to M's complaint by adding the keyword `synchronized` to the declaration of the `increase_love_count` method in `POOUser`. One day later, M comes back and has the same complaint again. Is it possible? Why or why not?

(4) (5%) Assume that the code of `POOUser` cannot be changed. How would you change the code of `POOLoveThread` to resolve the complaint of M?