Object-oriented programming, spring 2005.

Exam questions for Tuesday the 28th of June from 9-13.

The exam is scheduled to last 4 hours. The exam set consists of 5 questions which each have a number of sub questions. The points for each question are written next to it. It is possible to obtain a total of 100 points.

The exam is an open book exam; you are allowed to use text books, copies of the lecture slides, and other written material. If your answers depend on classes or methods from the textbook or lecture notes, you do not need to copy them, but give a precise reference with page number, section number etc.

You are not allowed the use of computers at the exam.

The exam set is 7 pages, including this front page.

1 Initialization order (15%)

This question is about object initialization. First sub-question is about initialization when there is no inheritance involved; second and third question involves inheritance as well. The question revolves around which values the field has during the different phases of object creation.

Consider the following class:

```
class A {
   int i = foo();
   A(int i){
      this.i = i + 10;
   }
   int foo(){ return 8; }
}
```

Question a) (6%). If we create an A-object as: `A a = new A(15);` – which values will the field `i` have from the object is allocated until the constructor has finished. In what order are the values assigned, and which rules in Java dictates this (you can use your own wording, or point to relevant paragraphs in the Java Precisely textbook).

Question b) (6%). Consider the following subclass of A (This class is legal Java and can compile):

```
class B extends A {
   int i = 55;
   B(int i){
      super(i+4);
      this.i = super.i + 3;
   }
   int foo(){ return i; }
}
```

If we create a B-object as: `B b = new B(15);` – which values will the fields of b have from the B object is allocated to when its constructor has finished, in what order, and by which rules?.

Question c) (3%). Assume A and B are declared as above. Will the following code print the same number twice, or different numbers? Why?

```
B b = new B(15);
A a = b;
System.out.println( a.i );
System.out.println( b.i );
```

2 Design by Contract and Iterator (15%)

In slide 5 for lecture 7 on design by contract and interfaces, a contract for iterators is given. In this question we will use this contract to implement our own iterator, which will merge the objects from two iterators into one iterator. The contract given below does not include the "remove" method. The remove method will not be part of this question.

The iterator contract is:

```
public interface Iterator {
    /* pre: none
    * post: return true if the iterator is not empty. */
    boolean hasNext();

    /* pre: hasNext()
    * post: return head of old.iterator & this is old.tail. */
    Object next();
}
```

The merging of two iterators a and b could be defined in many ways. Here we will assume that a and b are merged into c the following way:

- The merged iterator c has elements as long as one of a and b has more elements.
- The next element in c is
    - either an element from a or b, chosen randomly if both a and b has more elements
    - an element from a if b has no more elements
    - an element from b if a has no more elements

The class MergeIterators will be roughly like:

```
class MergeIterators implements Iterator{
    Iterator a,b;
    MergeIterators(Iterator a, Iterator b){
        ...
    }
    boolean hasNext(){...}
    Object next(){...}
}
```

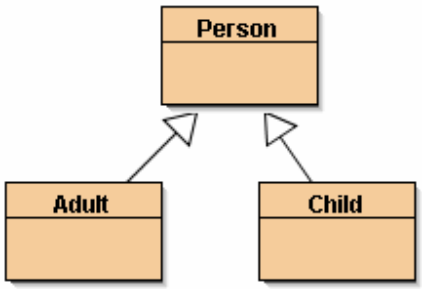
Question a). (5%). Give a class-invariant for MergeIterators. Use this invariant in coding the MergeIterators class in question b. *Notice, if you cannot figure out an invariant, just code* MergeIterators *in question b.*

Question b). (10%) Fill in the constructor, the hasNext and the next method. Hint. To choose randomly between a and b, the static method "double random()" defined in the class java.lang.Math will return a random number larger than or equal to 0, and less than 1.

3 Collections and generic classes (25%)

This question is about designing a collection class with generic parameters. If you cannot figure out how to use generic parameters, there might still be about 10% for you in here, so keep on.

The idea is that first we will design a class Kindergarden, which for this purpose in essence is a collection of Children. Next we will make a more general institution which can house any kind of person. We will in this question not look at the specific properties of Persons and Children. We will assume the following inheritance hierarchy



Question a) (5%) Write a class Kindergarden which has to methods - void enroll(Child c), which adds c to the set of children of the Kindergarden, and void withdraw(Child c), which removes c from the set of children of the Kindergarden.

Question b) (5%) Add methods to the class Kindergarden to implement the Iterable interface. This should allow us write code like:

```
Kindergarden k;
...
for(Child c: k) Santa.talkTo( c);
```

Question c) (8%) Rewrite class Kindergarden into class Institution, which takes a generic parameter stating which kind of Person the Institution is for. You can rewrite either the solution from a or b. Rewriting a) gives 5% Rewriting b) gives 8%.

Question d) (7%) Sometimes all the people at one institution will have to be moved to another institution – for instance when a small school is closed, and all the pupils have to move to another school. The merge method below is added to the Institution class, and does the necessary work.

```
public <…1…> void  merge(Institution<…2…> other){
      for(…3… person: other) {other.withdraw( person); this.enroll( person );}
}
```

There are three open spots regarding type parameters. What should be written each place and why? If you do not think <...1...> is necessary, leave it out.

4 Exception handling (20%)

This question is about exception handling. The fictious company CoolRUs develops temperature sensors for intelligent buildings. They have to develop a Java class which can be used for reading the temperature from within Java programs. You have just been hired for this company, and you are now to design part of this class, named TemperatureSensor. The class is in principle quite simple. There exist already a method prim_read(), which returns a double indicating the temperature in Celsius, but sometimes it returns -300 to indicate that the connection to the physical sensor is out of order. This normally happens rarely.

Question a) (5%) You must implement a method double readTemperature(). You can use the prim_read() method. You can choose to either just return -300 to indicate error, or you can throw an exception (checked or unchecked). What are the arguments for and against each strategy?

Question b) (5%) Independently of what you wrote in a), it is now decided that you should throw a checked exception. Write an outline of class TemperatureSensor which declares an Exception class SensorError, and which implements the readTemperature() method to throw this in case of failure.

Question c) (10%) Write a method monitorTemperature which monitors the temperature. The method has two arguments, low and high. It should read the temperature every second, and write out a warning (System.out.println) if the temperature is below low, or above high. If there is no response from the reader for one minute (it returns -300 every time it is queried within one minute), the monitorTemperature method should write out a special warning. *Hint. The static method 'void sleep(long millis) throws InterruptedException' in class Thread will pause for the number of milliseconds. InterruptedException is a checked exception.*

5 Observing behaviour (25%)

This question continues the story of the CoolRUs company and the class TemperatureSensor. The question can be solved independently of specific solutions to question 4. The question takes it's outset in the observer design pattern discussed in connection with the GUI lecture.

The monitorTemperature method from question 4.c printed warning messages to the screen. Suppose we want to do something more meaningful. As designers at CoolRUs, we cannot however, foresee all possible things to do when the temperature becomes out of range. This question is about designing a general temperature monitor. We will start out with the following class:

```
class GeneralTemperatureMonitor
   public interface TemperatureObserver {
      void observed(double temperature);
```

```
    void noRead();
  }

  public List<TemperatureObserver> observers
                        = new ArrayList<TemperatureObservers>();

  public void startMonitoring(final TemperatureReader tr, final long pause){
    Thread monitorThread = new Thread(){
      public void run(){
        while(true){
          try{ Thread.sleep(pause);}catch(InterruptedException ie){};
          try{
            double temp = tr.readTemperature();
            for(TemperatureObserver to:observers)
              to.observed( temp );
          }catch(TemperatureSensor.SensorError re){
            for(TemperatureObserver to:observers)
              to.noRead();
          }
        }
      }
    };
    monitorThread.start();
  }
}
```

The idea is that the user programs who are interested in temperature readings implements the TemperatureObserver interface, and add themselves to the observers list (*Notice that observers list is public*). The startMonitoring(…) method starts a new thread, which at regular intervals (the interval length is defined by the parameter pause) checks the temperature and reports its readings to the observers.

Question a) (9%) Write a class MyMonitor which in its constructor gets a reference to a GeneralTemperatureMonitor, and which adds itself as observer. When the temperature becomes below 15, it should print a warning, and when it becomes above 25 is should also print a warning. After 5 consecutive failures to read, it should print an error message. MyMonitor should not call startMonitor – this is taken care of by somewhere else.

Question b) (8%) We at CoolRus realize that one of the things all users write in their observed() method is a test to check the temperature reading. Only in case it fulfills certain criteria do they actually do something. We want to accommodate this usage pattern. Change the interface TemperatureObserver to include a method boolean checkReading(double temp). The idea is that the GeneralTemperatureMonitor should call this method with a concrete reading, and only call observed() if the checkReading method returns true. Change startMonitoring so that we only call observed under these new

circumstances. You need not hand in the entire startMonitoring method, just state what needs to be changed (E.g. *this text* should be replaced by *that text*).

Question c) (8%) Change solution a) so that it implements the new TemperatureObserver interface.