ESE112– Programming with Java

Quiz 3 – November 21, 2006

Name: ____________________________________________

Student ID: ______________________________________
(from your PennCard)

Email ___________________________________________ @seas.upenn.edu

Instructions:

• You have 50 minutes to answer all of the questions. The quiz is worth 60 points. The point value of each question is given.

• Write your answers on the quiz pages. The back side of each page may be used as a scratch pad.

• Questions during the quiz should be about the wording only. If you have a question, raise your hand and we’ll come to you. (This is less disruptive for others than if you come to us.)

• DON’T PANIC!. If you find a question that you cannot solve right away, consider moving on and returning to it after you finish the rest of the questions.

• Good luck!

<table>
<thead>
<tr>
<th>Score</th>
<th>Max</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>True/False Questions about Inheritance and Interfaces</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
<td>Inheritance Trees (Robot)</td>
</tr>
<tr>
<td>3</td>
<td>24</td>
<td>More Questions about Inheritance and Interfaces</td>
</tr>
<tr>
<td>Total</td>
<td>60</td>
<td></td>
</tr>
</tbody>
</table>
True/False Questions about Inheritance and Interfaces

1. (a) True or false: One of the advantages of inheritance is code reuse - code can be put in a class and used by any of its subclasses.  
   
2 points

(b) True or false: One of the advantages of inheritance is that it allows for polymorphism - code can be written for a class and used by any of its subclasses.  
   
2 points

(c) True or false: One of the advantages of interfaces is that it allows for polymorphism - code can be written for an interface and be used by any class that implements it.  
   
2 points

(d) True or false: Every Java object "is a" Object (in other words, every class inherits from the Object class).  
   
2 points

(e) True or false: Every object inherits a toString method from the Object class which returns the object’s heap address. However this method is typically overridden to return information about an object’s state, for debugging purposes.  
   
2 points

(f) True or false: An abstract class can not be instantiated. In other words, given an abstract class X, Java will not allow the operation:  
   
new X()  
   
2 points

(g) True or false: An interface typically consists of a few abstract methods (methods without bodies).  
   
2 points

(h) True or false: A class can extend (inherit from) at most one class, but it can implement multiple interfaces  
   
2 points

(i) Assume that the following code is valid (it compiles and runs):  
   
   Component c = new Component();  
   c.process();  

True or false: The JVM (Java Virtual Machine) looks for a method called process() in the Component class. If found, it is executed. Otherwise it looks for the method in the parent class, grandparent class, greatgrandparent class, and so on, until it finds one to execute.  

2 points

(j) True or false: A concrete class X that has an abstract method in its ancestry (either in an abstract class or interface Y) must have a concrete version of the method either directly in the concrete class X or in an ancestor of X which is a subtype of Y.  

2 points
Inheritance Trees

2. Draw an inheritance tree for the following software system:

Concrete classes RescueBot and FactoryBot inherit from abstract class Robot. Robot implements the interfaces Displayable and Programmable. RescueBot implements the interface RemoteControllable. SearchAndRescueBot is a concrete class that is a special kind of RescueBot. Include the Object class in your diagram. Use rectangles for concrete classes, ovals for abstract classes, and triangles for interfaces. Within the rectangle/oval/triangle write the name of the class/interface AND "concrete class"/"abstract class"/"interface. (We ask for this last part because it is sometimes hard to tell if people have drawn a rectangle, oval, or triangle.) If you think there is ambiguity in the directions, make a decision and briefly justify it.
More Questions about Inheritance and Interfaces

3. Answer the following questions. Each question should be considered individually (in other words X in one problem is not the same as X in another problem).

(a) Given that this code is valid:
    ```java
    new X()
    ```
    We can infer that X could be a (circle all possible choices):
    
    concrete class  abstract class  interface

(b) Given that this code is valid:
    ```java
    X thing; // a variable declaration statement
    ```
    We can infer that X could be a (circle all possible choices):
    
    concrete class  abstract class  interface

(c) Given that this code is valid:
    ```java
    public void foo(X thing){
    ...  
    }
    ```
    We can infer that X could be a (circle all possible choices):
    
    concrete class  abstract class  interface

(d) Given that this code is valid:
    ```java
    public void foo(X thing){
    ...  
    }
    ```
    Things that may be passed as an argument to the foo method may have what type (circle one choice)?

    type X only
    X or a subtype of X
    X or a supertype of X
    X, a subtype, or a supertype of X
(e) Given that this code is valid:

```java
X thing = new Y();
```

What can we infer about the relationship between X and Y?

(f) Abstract methods are found in abstract classes and interfaces. The rule is that a concrete class (a class that can be instantiated with the "new" operator) that has an abstract method in its ancestry must have a corresponding concrete method either directly inside it or in an ancestral class. (For example, in a hypothetical Animal hierarchy, Lion and SeeingEyeDog both implement the Trainable interface which has abstract methods `sit()` and `stand()`. The Lion class has `sit()` and `stand()` methods. SeeingEyeDog’s parent class, Dog, has `sit()` and `stand()` methods.)

Why bother having the Trainable interface? What advantages does it provide?