CS 2510 Exam 1 – Fall 2009

Name: _____

Student Id (last 4 digits): _____

• Write down the answers in the space provided.
• You may use all forms that you know from *ProfessorJ (Beginner)*, or *ProfessorJ (Intermediate)*, where indicated. If you need a method and you don't know whether it is provided, define it. You do not need to include the curly braces for every `if` or every `else`, as long as the statements you write are correct in standard Java.
• For tests you only need to provide the expression that computes the actual value, connecting it with an arrow to the expected value. For example `s.method() -> true` is sufficient.
• Remember that the phrase "develop a class" or "develop a method" means more than just providing a definition. It means to design them according to the **design recipe**. You are *not* required to provide a method template unless the problem specifically asks for one. However, be prepared to struggle if you choose to skip the template step.
• We will not answer *any* questions during the exam.

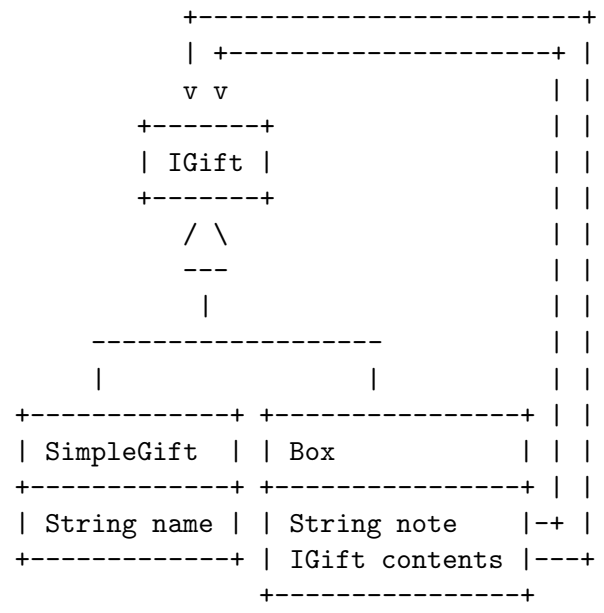
Good luck.

Problem	Points	/
1		/24
Total		/24

21 Points

Problem 1

Here is a Java class diagram that describes a gift you may receive for your birthday:



A. Write down the Java class and interface definitions that are represented by this class diagram.

Solution [POINTS 3: 1 point for the interface, 1 point for the class `SimpleGift`, 1 point for the class `Box`]

```
// to represent a gift
interface IGift{}

// to represent a simple gift
class SimpleGift implements IGift{
  String name;

  SimpleGift( String name){
     this.color = name;
  }
}

// to represents a gift in a box with a note
class Box implements IGift{
  String note;
  IGift contents;

  Box(String note, IGift contents){
     this.note = note;
     this.contents = contents;
  }
}
```

B. Make examples of two gifts, a simple gift, and a gift enclosed in at least two `Box`es.

Solution [POINTS 2: one point for the `SimpleGift`, and one point for the example with two `Box`es.]

```
IGift simple = new SimpleGift("book");
IGift twoBoxes = new Box("Happy birthday",
                         new Box("Love, mom",
                                 new SimpleGift("chocolate")));
```

C. Design the method `countBoxes` that counts the number of boxes that the gift is packaged in.

Solution [POINTS 5: 1 point purpose/header; 1 point body in each class; 2 points examples – should include result 0 and result > 1]

```
// in the interface IGift:
  // count the number of boxes this gift is contained in
  int countBoxes();

// in the class SimpleGift:
  // count the number of boxes this simple gift is contained in
  int countBoxes(){
    return 0;
  }

// in the class Box:
  /* TEMPLATE:
    ... this.note ...                    -- String
    ... this.contents ...                -- IGift

    ... this.contents.countBoxes() ...   -- int
  */

  // count the number of boxes this gift is contained in
  int countBoxes(){
    return 1 + this.contents.countBoxes();
  }

// in the class Examples:
IGift simple = new SimpleGift("book");
IGift twoBoxes = new Box("Happy birthday",
                         new Box("Love, mom",
                                 new SimpleGift("chocolate")));

  // test the method countBoxes in the classes that represent a gift
  boolean testCountBoxes(){
    return (check this.simple.countBoxes() expect 0) &&
           (check this.twoBoxes.countBoxes() expect 2);
  }
```

D. You really like chocolate and want to know whether this gift (its `name`) is `"chocolate"`. Design the method `isName` that tells us whether the name of the gift is what we hope it is. In your examples, include tests that determine whether some gift contains `"chocolate"`.

Solution [POINTS 5: 1 point purpose/header; 1 point body for the `SimpleGift` class, 1 point body for the `Box` class, 2 points for examples for the `isName` method.]

```
// in the interface IGift:
  // is the name of this gift the given name?
  boolean isName(String name);

// in the class SimpleGift:
  // is the name of this simple gift the given name?
  boolean isName(String name){
    return this.name.equals(name);
  }

// in the class Box:
  // is the name of this boxed gift the given name?
  boolean isName(String name){
    return this.contents.isName(name);
  }

// in the class Examples:
IGift simple = new SimpleGift("book");
IGift twoBoxes = new Box("Happy birthday",
                         new Box("Love, mom",
                                 new SimpleGift("chocolate")));

  // test the method isName in the classes that represent a gift
  boolean testCountBoxes(){
    return (check this.simple.isName("book") expect true) &&
           (check this.simple.isName("chocolate") expect false) &&
           (check this.twoBoxes.isName("book") expect false) &&
           (check this.twoBoxes.isName("chocolate") expect true);
  }
```

E. You are curious about all the notes that came in the boxes that wrapped your gift. Design the method `getNotes` that combines into one `String` all notes that came with this gift. Of course, it may happen that there were no notes attached to the gift, in which case your method produces just the empty `String ""`.

Solution [POINTS 5: 1 point purpose/header; 1 point body for the `SimpleGift` class, 1 point body for the `Box` class, 2 points for examples for the `getNotes` method.]

```
// in the interface IGift:
  // produce the complete note that came with this gift.
  String getNote();

// in the class SimpleGift:
  // produce the complete note that came with this simple gift.
  String getNote(){
    return "";
  }

// in the class Box:
  // produce the complete note that came with this boxed gift.
  String getNote(){
    return this.note.concat(this.contents.getNote());
  }

// in the class Examples:
IGift simple = new SimpleGift("book");
IGift twoBoxes = new Box("Happy Birthday ",
                         new Box("Love, mom",
                                 new SimpleGift("chocolate")));

  // test the method isName in the classes that represent a gift
  boolean testCountBoxes(){
    return (check this.simple.getNote() expect "") &&
           (check this.twoBoxes.getNote()
                  expect "Happy Birthday Love, mom");
  }
```

F. Show the templates for all classes in this problem for which you have designed methods.

Solution [POINTS 4: 1 point template for `SimpleGift`, 3 points template for `Box`: 1 point for fields, 1 point for methods for contents, 1 point for data types]

```
// in the class SimpleGift
  /* TEMPLATE:
    ... this.name ...                  -- String

    ... this.countBoxes() ...          -- int
    ... this.isName(String) ...        -- boolean
    ... this.getNotes() ...            -- String
  */

// in the class Box
  /* TEMPLATE:
    ... this.note ...                      -- String
    ... this.contents ...                  -- IGift

    ... this.contents.countBoxes() ...     -- int
    ... this.contents.isName(String) ...   -- boolean
    ... this.contents.getNotes() ...       -- String
  */
```