CS 2510 Exam 1 – Fall 2010

Name: ____________________

Student Id (last 4 digits): ____________________

• Write down the answers in the space provided.
• You may use all syntax that you know from *FunJava* other than *abstract classes*. 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 otherwise correct in FunJava.
• 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 "design a class" or "design 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	/
A		/ 0
B		/ 4
C		/10
D		/ 8
E		/ 4
Total		/26

25 Points

Problem 1

Here is a Java class diagram that describes a maze with treasures to hunt for:

```
                  +---------------------------------+
                  | +------------------------------+|
                  | | +---------------------------+||
                  | | |                           |||
                  v v v                           |||
                +-------+                         |||
                | IMaze |                         |||
                +-------+                         |||
                   / \                            |||
                   ---                            |||
                    |                             |||
       --------------------------------           |||
       |               |              |           |||
+--------------+ +------------+   +-------------+ |||
| Goal         | | Tunnel     |   | Room        | |||
+--------------+ +------------+   +-------------+ |||
| String prize | | IMaze next |-+ | IMaze left  |-+||
| int value    | | int length | | | IMaze right |--+|
+--------------+ +------------+ | +-------------+   |
                                +-------------------+
```

A. *(0 points)*

You are not required to do this. You may want to see what the data definitions look like. Do not spend much time on this, unless you do not understand what the data represents. The examples in the next part should make that clear.

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

____________**Solution** ____________ [POINTS 0: 1 point for the interface, 1 point for the class `Goal`, 1 point for the class `Tunnel`, 1 point for the class `Room`; subtract 1 point if there are no purpose statements]

```
// to represent a maze in a game
interface IMaze{}

// to represent the goal reached in a maze game
class Goal implements IMaze{
  String prize;
  int value;

  Goal(String prize, int value){
    this.prize = prize;
    this.value = value;
  }
}

// to represent a tunnel in a maze game
class Tunnel implements IMaze{
  IMaze next;
  int length;

  Tunnel(IMaze next, int length){
    this.next = next;
    this.length = length;
  }
}

// to represent a room in a maze game
```

```
class Room implements IMaze{
  IMaze left;
  IMaze right;

  Room(IMaze left, IMaze right){
    this.left = left;
    this.right = right;
  }
}
```

B. *(4 points)*

The following picture represents maze that can be represented by this data definition. Write down the data definitions needed to represent the information in this picture.

```
                                             /-------\
                                            | Gold:20 |
                                             \-------/
                     +---| |---+            +---| |---+
 /---------\         |         |            |         |
| Diamond:40 *T1:20**   Room1   **T2:50*****   Room2  |
 \---------/         |         |            |         |
                     +---------+            +---| |---+
                                     /-----\|         |/---------\
                                    | Lead:0   Room3    Silver:10 |
                                     \-----/|         |\---------/
                                            +---------+
```

____________**Solution** ____________ [POINTS 4: 1 point for the class `Goal`, 1 point for the class `Tunnel`, 2 points for the class `Room`]

```
class ExamplesMaze{
  ExamplesMaze(){}

  IMaze diamond = new Goal("Diamond", 40);
  IMaze gold = new Goal("Gold", 20);
  IMaze silver = new Goal("Silver", 10);
  IMaze lead = new Goal("Lead", 0);

  IMaze r3 = new Room(this.silver, this.lead);
  IMaze r2 = new Room(this.gold, this.r3);

  IMaze t1 = new Tunnel(this.diamond, 20);
  IMaze t2 = new Tunnel(this.r2, 50);

  IMaze r1 = new Room(this.t1, this.t2);
}
```

C. *(10 points)* Design the method `valueOf` that determines the value of the prize described by a given `String` in this maze. There may be several prizes with the same name, in that case add up their values.

______________**Solution** ______________ [POINTS 10: 1 point purpose/header; 5 points bodies in each class 2-1-2); 4 points examples – should include each class and test the decision in Goal]

```
// in the interface IMaze:
  // what is the value of the given prize in this maze?
  int valueOf(String prize);

// in the class Goal:
  // what is the value of the given prize in this maze?
  int valueOf(String prize){
    if (prize.equals(this.prize)){
      return this.value;}
    else{
      return 0;}
  }

// in the class Tunnel:
  // what is the value of the given prize in this maze?
  int valueOf(String prize){
    return this.next.valueOf(prize);
  }

// in the class Room:
  // what is the value of the given prize in this maze?
  int valueOf(String prize){
    return this.left.valueOf(prize) +
           this.right.valueOf(prize);
  }

// in the class Examples:
  // test the method valueOf for the maze classes
  boolean testValueOf(Tester t){
    return
    t.checkExpect(this.gold.valueOf("Gold"), 20) &&
    t.checkExpect(this.t2.valueOf("Gold"), 20) &&
    t.checkExpect(this.t1.valueOf("Gold"), 0) &&
    t.checkExpect(this.r0.valueOf("Gold"), 50);
  }
```

D. *(8 points)*

Design the method `longestPath` that computes the longest path in a maze. The lengths of the tunnels are given (in feet), each room is 10 feet long and 10 feet wide.

_____________**Solution**_____________ [POINTS 8: 1 point purpose/header; 1 point body for the `Goal` class, 1 point body for the `Tunnel` class, 2 points body for the `Room` class, 3 points for examples for the `longestPath` method.]

```
// in the interface IMaze:
  // compute the length of the longest path in this maze
  int longestPath();

// in the class Goal:
  // compute the length of the longest path in this maze
  int longestPath(){
    return 0;
  }

// in the class Tunnel:
  // compute the length of the longest path in this maze
  int longestPath(){
    return this.length + this.next.longestPath();
  }

// in the class Room:
  // compute the length of the longest path in this maze
  int longestPath(){
    return 10 + this.max(this.left.longestPath(),
                         this.right.longestPath());
  }

  // find the maximum of two given integers
  int max(int a, int b){
    if (a > b){
      return a;}
    else{
      return b;}
  }

// in the class Examples:

  // test the method longestPath for the maze classes
```

```
boolean testLongestPath(Tester t){
  return
  t.checkExpect(this.gold.longestPath(), 0) &&
  t.checkExpect(this.t2.longestPath(), 70) &&
  t.checkExpect(this.t1.longestPath(), 20) &&
  t.checkExpect(this.r1.longestPath(), 80);
}
```

... This page is intentionally left blank ...

E. *(4 points)*

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

____________**Solution**____________ [POINTS 4: 1 point template for `Treasure`, 3 points template for `Room`: 1 point for fields, 1 point for methods for fields, 1 point for data types]

```
// in the class Goal
TEMPLATE:
  FIELDS:
  ... this.prize ...              -- String
  ... this.value ...              -- int

  METHODS:
  ... this.valueOf(String) ...    -- int
  ... this.longestPath() ...      -- int

  METHODS FOR FIELDS:


// in the class Tunnel
 TEMPLATE:
  FIELDS:
  ... this.next ...              -- IMaze
  ... this.length ...            -- int

  METHODS:
  ... this.valueOf(String) ...    -- int
  ... this.longestPath() ...      -- int

  METHODS FOR FIELDS:
  ... this.next.valueOf(String) ...    -- int
  ... this.next.longestPath() ...      -- int


// in the class Room
 TEMPLATE:
  FIELDS:
  ... this.left ...              -- IMaze
... this.right ...             -- IMaze

METHODS:
... this.valueOf(String) ...    -- int
... this.longestPath() ...      -- int

METHODS FOR FIELDS:
... this.left.valueOf(String) ...    -- int
... this.left.longestPath() ...      -- int

... this.right.valueOf(String) ...    -- int
... this.right.longestPath() ...      -- int
```