STUDENT NAME: ______________________________

CSE331 Autumn 2011 Midterm Examination

October 28, 2011

- 50 minutes; 75 points total.
- Open note, open book, closed neighbor, closed anything electronic (computers, web-enabled phones, etc.)
- An easier-to-read answer makes for a happier-to-give-partial-credit grader

	Points	Your score
Part I: T/F and explain	12	
Part II: Testing	22	
Part III: Specification	18	
Part IV: ADTs	20	
Part V: Miscellaneous	3	
Total	**75**	

Don't turn the page until the proctor gives the go ahead!

STUDENT NAME: ______________________________

Part I: True/False with a one-sentence justification (12 points total)

1. A representation invariant for an ADT implementation must hold before and after each statement in every method of that implementation.

2. Consider the `Duration` class presented in Lecture #4:

```
public class Duration {
  private final int min;
  private final int sec;
  public Duration(int min, int sec) {
     this.min = min;
     this.sec = sec;
  }
}
```

   True or false: The following `HashCode` definition is safe for instances of the `Duration` class:

```
public int hashCode() {
   return sec*min;
}
```

3. If all constructors of a Java class **C** are declared to be **`private`**, no client of the class can create objects of type **C**.

4. Extending a Java abstract class can sometimes produce a true subtype.

5. When a JUnit test fails, the programmer must change the program being tested.

6. Java generics shift the time at which programmers see errors from run-time to compile-time.

STUDENT NAME: ____

Part II: Testing (22 total points)

1. (6 points) A basic implementation of `getGreeting` from `RandomHello` is:

```
/**
 * @return a random greeting from a list of
 * five different greetings.
 */
public static String getGreeting() {
  Random randomGenerator = new Random();
  return(greetings[randomGenerator.nextInt(5)]);
}
```

If you replace the last statement with

```
return(greetings[(randomGenerator.nextInt(5)+103312)%5])
```

where `%` is the modulus operator (`x % y` means `x mod y`), which of the following tests (discussed in lecture and having descriptive names), if any, fail for this implementation? Briefly justify.

- `testDoes_getGreeting_returnDefinedGreeting`
- `testDoes_getGreetingNeverReturnSomeGreeting`
- `test_UniformGreetingDistribution`

2. (8 points) Consider a simple form of random test generation, where the data passed to a method is selected randomly. For example, for **`binarySearch`** the length and contents of the array to be searched can be selected at random, as can the key to search for. Describe (in at most two sentences for each) two challenges that would arise from this simplistic method of testing.

3. (8 points) Consider the following specification that is a variant of the binary search specification – basically, it now takes an unsorted array:

```
/**
 * @param data is an int array in which to search for the key
 * @param key the key to search for in data
 * @returns some i such that data[i] = key if such an i exists,
 *          otherwise -1
 */
public static int unsortedSearch(int[] data, int key)
```

Write four black-box tests for this method. Each test should be intended to exercise a different possible subdomain, and you must briefly describe what that intended subdomain is for each test. For example, here is one test (no, you cannot repeat this one) and description:

```
@Test
public void testFoundKeyNearMiddle() {
  int[] td = {9, 32, -18, 99, 77};
  assertEquals(2,UnsortedSearch.unsortedSearch(td,-18));
}
```

Description: This is a basic test of the subdomain representing success in finding a key in the middle of the array.

Part III: Specifications (15 total points)

1. (10 points) Consider the following code (which compiles, runs, etc.):

```
static void uniquify(List<Integer> lst) {
  for (int i=0; i < lst.size()-1; i++)
    if (lst.get(i) == lst.get(i+1))
      lst.remove(i);
}
```

Fill-in the following specification for this code:

```
static void uniquify(List<Integer> lst)
  requires ???
  modifies ???
  effects  ???
  returns  ???
```

STUDENT NAME: ____

2. (8 points) Is the following situation possible (**S** means specification, **I** means implementation)?

   S_1 is satisfied by I_1 $\wedge$ S_2 is satisfied by I_1 $\wedge$
   S_1 is satisfied by I_2 $\wedge$ S_2 is satisfied by I_2 $\wedge$
   $S_1 \neq S_2$

   Briefly justify your answer.

STUDENT NAME: ____

Part IV: ADTs (20 total points)

1. (12 points total) For each of the four following pairs of ADTs, select
   - **`T1< T2`**: **`T1`** is a true subtype of **`T2`**
   - **`T1> T2`**: **`T2`** is a true subtype of **`T1`**
   - **`T1 ≠ T2`**: **`T1`** and **`T2`** are incomparable
   - **Other**: Cannot tell from this information

	< > ≠ Other? [Write answers in this column]	**T1**	**T2**
(a)		**2-dimensional points** [Standard mutable 2D (x,y) points and operations]	**3-dimensional points** [Standard mutable 3D (x,y,z) points and operations]
(b)		**Java `Integer`**	**Java `int`**
(c)		**Java `String`**	**Java `Character[]`**
(d)		**`AbstractCollection<E>`** [`AbstractCollection<E>` provides a skeletal implementation of the `Collection` interface, to minimize the effort required to implement this interface.]	**`Set<E>`** [The `Set` interface places additional stipulations, beyond those inherited from the `Collection` inter-face, on the contracts of all constructors and on the contracts of the `add`, `equals` and `hashCode` methods.]

STUDENT NAME: ____

2. (8 points total) Consider the following Java code (which is legal and not intended to be surprising or confusing):

```
public class DayHour {
  private int day;
  private int hour;

  public DayHour(int d, int h) { // constructor
    day = d;
    hour = h;
  }

  public int getDay() {
    return day;
  }

  public int getHour() {
    return hour;
  }

  public String getHour12() {
    if (hour > 12)
      return hour + "AM";
    else
      return (hour - 12) + "PM";
  }
}
```

   a. (4 points) What is the most appropriate representation invariant for the class?

   b. (4 points) If the class has some representation exposure, briefly describe it. If not, briefly justify why not.

STUDENT NAME: ____

Part V: Miscellaneous (3 total points)

What part of the course so far has taken the most time "stuck" on something? A particular Java language construct or concept? A particular function of Eclipse or of JUnit? In other words, what have you spent the most time on for the least value in the course?