CS 201, Summer 2015 — June 4th — Exam 1 Name: _____

Question 1. [6 points] Write statements to read two `double` values from the user, compute their sum, and then print the sum.

Question 2. [3 points] Fill in the blank: a(n) _____ exception must be dealt with using `try/catch` or by using a `throws` declaration to allow the method to throw the exception out of the method.

Question 3. [3 points] Fill in the blank: a(n) _____ exception does not need to be dealt with using `try/catch` or by using a `throws` declaration to allow the method to throw the exception out of the method.

Question 4. [3 points] Fill in the blank: the _____ construct is useful for guaranteeing that cleanup actions (such as closing a `FileReader`) are executed even if an exception is thrown.

Question 5. [3 points] Briefly explain the circumstances under which it is appropriate to throw an exception out of a method rather than handling it using `try/catch`.

Question 6. [3 points] Briefly explain the circumstances under which it is appropriate to handle an exception using `try/catch` rather than throwing it out of the method.

Question 7. [6 points] For each of the following C terms, state the closest equivalent Java term:

C term	Java term
Function	
Struct type	
Struct instance	

Question 8. [6 points] What output is printed by the following program (which begins on the left and continues on the right)?

```
public class Box {
  private int value;

  public Box(int v)
    { value = v; }
```

```
  public static void main(
      String[] args) {
    Box b1 = new Box(17);
    Box b2 = new Box(42);

    System.out.println(b1.value);
    System.out.println(b2.value);
  }
}
```

Question 9. [6 points] What output is printed by the following program (which begins on the left and continues on the right)?

```
public class Box2 {
  private int value;

  public Box2(int v)
    { value = v; }
```

```
  public static void main(
      String[] args) {
    Box2 b1 = new Box2(17);
    Box2 b2 = b1;

    b2.value = 42;

    System.out.println(b1.value);
    System.out.println(b2.value);
  }
}
```

Question 10. [6 points] What output is printed by the following program (which begins on the left and continues on the right)?

```
public class Mystery {
  public static void f(
      int[] arr) {
    arr[0] = arr[1];
  }
```

```
  public static void main(
      String[] args) {
    int[] nums = new int[3];
    nums[0] = 62;
    nums[1] = 84;
    nums[2] = 4;
    f(nums);
    System.out.printf("%d %d %d\n",
      nums[0], nums[1], nums[2]);
  }
}
```

Question 11. [5 points] Briefly explain the roles of the superclass and subclass in an inheritance ("Is-A") relationship.

Question 12. [10 points] Show the code for a class called `Adder` as described below. An `Adder` object should store a single `int` value. The class should have a constructor which allows a new `Adder` object's value to be initialized to a specified value. The class should have a `getValue` method which returns the `Adder`'s current value. The class should have an `add` method which takes an integer value as a parameter and adds it to the `Adder`'s current value.

The following JUnit tests show the expected behavior:

```
Adder a1 = new Adder(0);
Adder a2 = new Adder(16);
assertEquals(0, a1.getValue());
assertEquals(16, a2.getValue());
a1.add(9);
assertEquals(9, a1.getValue());
a1.add(5);
assertEquals(14, a1.getValue());
a2.add(3);
assertEquals(19, a2.getValue());
a2.add(10);
assertEquals(29, a2.getValue());
```

Programming Question

To get started, use a web browser to download the zipfile as specified by your instructor. Import it as an Eclipse project using File → Import... → General → Existing Projects into Workspace → Archive file.

Important: You may use the following resources:

- The textbook
- The lecture notes posted on the course web page
- Your previous labs and assignments

Do not open any other files, web pages, etc.

Question 13. [40 points] Complete the implementation of the `ComboLock` class. This class represents a combination lock, and should have each method described below.

Constructor: Takes three integer arguments, representing the numbers in the `ComboLock`'s combination.

`spin`: "Spins" the `ComboLock`. Takes a single integer parameter. If `spin` is called three times, and the sequence of values passed match the combination set when the `ComboLock` object was created, then the lock is *unlocked*.

`isUnlocked`: Checks whether the `ComboLock` is unlocked (as explained in the description for the `spin` method above). Returns `true` if the `ComboLock` is unlocked, or `false` if it is locked. This method does not take any parameters.

`lock`: "Resets" the combination lock by negating the effects of any previous calls to `spin`. In other words, after `lock` is called, the only way to unlock the `ComboLock` is to call `spin` three times, passing the sequence of numbers matching the combination. This method does not take any parameters.

Hints and requirements:

- In addition to adding the methods described above, you will need to add fields to keep track of the state of the `ComboLock`.
- The `ComboLockTest` JUnit test class can be used to test your implementation. Note that this class doesn't test the `lock` method: you should write at least one test for this method to make sure your `lock` method works correctly.
- Make sure that the `isUnlocked` method returns true only when the `spin` method has been called three times (following the creation of the `ComboLock` or a call to `lock`), passing the sequence of numbers matching the combination.

When you are ready to submit your code, select the **CS201_Exam01** project in the package explorer, and click the blue up arrow button in the toolbar. Enter your Marmoset username and password when prompted.