CSC150 Midterm Exam

October 15, 2003

Instructor: Steve Engels

Last Name: _____

First Name: _____

Student Number: _____

Instructions:

- **Do NOT open the exam until the examiner has told you to do so**.
- Once you receive the signal to start, fill in your name and student number on all of the pages of this exam.
- The question sheets for Q2 and Q3 are on the last two pages of the exam. Tear them off and keep them on the side before starting these questions.
- There are no aids allowed. All bags and personal belongings must be placed at the front of the room.
- Comments are not required except where indicated, although they will help us mark your answer. They may also give you part marks if you cannot figure out how to write the code for a question.
- Write clearly and neatly. The markers will not be required to decipher illegible handwriting

Q1: _____ / 20

Q2: _____ / 20

Q3: _____ / 10

Total: _____ / 50

Short Java API Descriptions (all methods are public):

```
class Math:
    static int max(int a, int b)  // returns the greater of two int values
    static int min(int a, int b)  // returns the lesser of two int values
    static int round(double a)  // returns the closest int to the argument
class String:
    String substring(int i, int j)  // returns the letters between position i
                                    // (inclusive) and position j (non-inclusive)
    String substring(int i)  // returns the letters from i (inclusive) to the end
    int indexOf(String s)  // returns the position of s in this String, -1 if not found
    int indexOf(String s, int i)  // returns the position of s in this String after
                                  // index i, -1 if not found
    int length()  // the number of letters in this String
```

Question #1: Multiple Choice / Short Answer (20 marks total)

a) What are two main **conceptual** differences between interfaces and abstract classes? In the boxes below, describe how abstract classes and interfaces differ in each case. Note: the `abstract, extends` and `implements` keywords are <u>not</u> concepts. (4 marks)

	Abstract Classes	**Interfaces**
Difference #1	Abstract classes can have variables declared and some methods implemented	All methods in an interface must be abstract, with no variables allowed
Difference #2	Abstract classes describe behaviour that must be extended by subclasses	Interfaces can be implemented by any class within the inheritance hierarchy

b) Write code to declare and initialize an array of 3 separate `Date` objects (2 mark)

Either: `Date[] d = new Date[3];` or `Date d[] = new Date[3];`

```
Then:    d[0] = new Date();
         d[1] = new Date();
         d[2] = new Date();
```

c) Write a one-line statement that implements the following method. Do not use if statements or loops. (4 marks)

```
/**
 *   getMiddleValue returns the middle value of
 *   the three integers given as input
 */
int getMiddleValue(int a, int b, int c)
{

return Math.max(Math.min(Math.max(a,b), c), Math.min(a,b));
  or
return Math.min(Math.min(Math.max(a,b), Math.max(b,c)), Math.max(a,c));

}
```

Circle any comments that would appear in a JavaDoc API. (1 mark)

i) `/* this kind */`

ii) `// this kind`

iii) `/** this kind */`

iv) `/** this kind **/`

d) Only one `assertEquals` statement takes in three parameters. Which one is it, and what is the third parameter for? Keep your answer brief. (2 marks)

There are multiple `assertEquals` methods that take in three parameters, but the one that I'm looking for is for `double`s.

The third parameter for this `assertEquals` is a range value, representing the maximum difference two doubles may have and still be considered "equal".

e) Write a method called `isDivisible` that takes in two integers, `big` and `little`, and returns `true` only if `little` divides `big` evenly (no remainder). There should only be one statement inside the method. You can assume that `little` is not zero. (3 marks)

```
public boolean isDivisible(int big, int little)
{
    return big % little == 0;
}
```

f) Assume that two integers named `count` and `index` have already been declared. Rewrite the following code to use an empty `while` loop instead: (4 marks)

```
for (count=1;  count<21; count++)
    if (count - index == 0)
        break;
```

```
count = 1;
while (++count < 21 && count - index != 0);

or

count = 1;
while ( ++count < 20 && count - index != 0);
```

Question #2

Write your answer to Question #2 here.

```java
public class Dog extends Animal
{

  private String breed;

  public Dog(String name, int age, String breed)
  {
    super(name, age);
    this.breed = breed;
  }

  public Dog(String name, int age)
  {
    super(name, age);
    breed = "no breed";
  }

  public String getBreed()
  {
    return breed;
  }

  public void speak()
  {
    if (breed.equals("no breed"))
      System.out.println("I have no breed, but " + super.toString());
    else
      System.out.println("I am a dog, and " + super.toString());
  }

}
```

Question #3

a) Write your answer to Question 3a here. **(5 marks)**

```
> Brillig b1 = new Brillig("lewis");


    lewis is empty.
    lewis count is 1.


> Brillig b2 = new Brillig("lewis");


    lewis count is 1.
    lewis count is 2.


> b1 == b2

    false

> b1.equals(b2)

    false
```

b) Write your answer to Question 3b here. **(5 marks)**

```
> Brillig b = new Slithy("lewis");


    slithy lewis is empty.
    slithy lewis count is 1.
    lewis count is -1.
    lewis is empty.




> b.toString()

    "Slithy lewis with 1 items."

> b

    Slithy lewis with 1 items.
```

Question #2: Class Design and Inheritance (20 marks total)

```
public abstract class Animal
{

  private String name;
  private int age;

  public Animal(String name, int age)
  {
    this.name = name;
    this.age = age;
  }

  public abstract void speak();

  public String toString()
  {
    return "my name is " + name;
  }

}
```

```
public class Beagle extends Dog
{

  private double price;

  public Beagle(String name, int age, double price)
  {
    super(name, age, "beagle");
    this.price = price;
  }

  public void speak()
  {
    System.out.println("I'm a " + getBreed() +
                         ", and " + super.toString());
  }

  public String toString()
  {
    return "Good grief, " + super.toString();
  }
}
```

Given the classes above, design the Dog class on Page 4 such that the following statements, when typed into DrJava's interactions pane, will generate the corresponding results. You may not add to either of the classes above, and information hiding principles should be upheld in your design. (16 marks)

Interactions

```
> Dog d = new Dog("Grimm", 10);
> d.speak();
I have no breed, but my name is Grimm
> d = new Dog("Lassie", 20, "collie");
> d.speak()
I am a dog, and my name is Lassie
> Beagle b = new Beagle("Snoopy", 25, 50);
> b.speak()
I'm a beagle, and my name is Snoopy
> b
Good grief, my name is Snoopy
>
```

Question #3: Program Tracing (10 marks total)

Given the code from this page, write the output that would appear in the interactions pane, as a result of the commands on Page 5. Assume that the interactions pane is reset before each part. Fill in the spaces provided, and make sure to write legibly.

Note: If you get stuck, drawing diagrams is a highly recommended technique for unsticking yourself.

```java
public class Brillig
{
  private static int totalCount = 0;
  private int count = 0;
  private String name;


  public Brillig(String name)
  {
    this.name = name;
    testCount();
    totalCount++;
    testCount();
  }

  private void testCount()
  {
    if (totalCount == 0)
      System.out.println(name + " is empty.");
    else
      System.out.println(name + " count is " + totalCount + ".");
  }

  public String toString()
  {
    return "Brillig " + name + " with " + count++ + " items.";
  }
}
```

```java
public class Slithy extends Brillig
{
  private static int totalCount = -1;
  private int count = 0;
  private String name;

  public Slithy(String name)
  {
    super("slithy " + name);
    this.name = name;
    testCount();
    totalCount++;
    testCount();
  }

  private void testCount()
  {
    if (totalCount == 0)
      System.out.println(name + " is empty.");
    else
      System.out.println(name + " count is " + totalCount + ".");
  }

  public String toString()
  {
    return "Slithy " + name + " with " + ++count + " items.";
  }
}
```