Final Examination
CS170: Introduction to Computer Science

Observe the Emory College Honor Code while taking this test.

Question 1. (30 pts, 3 each) Multiple Choice

For each question, circle the best option.

1.1. Which of the following statement is true concerning a safe casting operation:

   - You must tell the compiler that you agree to a loss of (some) information.
   - It is performed by the Operating System.
   - It is an automatic conversion.
   - It may cause a program error.
   - None of the above.

1.2. What mechanism is used to define a new class using an existing class as basis:

   - Shadowing
   - Inheritance
   - Initialization
   - Construct
   - None of the above.

1.3. If a and b are int variables and b is not zero, which of the following expressions equals a - (a/b)*b:

   - 0
   - a - b
   - a & b
   - a / b
   - None of the above.

1.4. Consider the following code fragment:

   ```
   int sum = 0;
   int item = 0;
   do
   {
       item++;
       sum += item;
       if ( sum > 4 )
           break;
   } while ( item < 5 );
   ```
What is the value of the variable sum after the loop has terminated?

- 6
- 10
- 15
- it is an infinite loop
- None of the above.

1.5. Which of the following statement is true about instance methods:

- Instance methods are defined using the keyword static.
- Instance methods are defined using the keyword void.
- Instance methods are defined without using the keyword static.
- Instance methods always have public access.
- None of the above.

1.6. Which of the following statement is true about constructor methods:

- Java always defines the default constructor in a user-defined class.
- A constructor method must declare void as its return type.
- When an object is created, Java always invokes a constructor method.
- A constructor method is always static.
- None of the above.

1.7. Suppose a class variable x (defined using the keyword static) in class myClass is shadowed by a local variable. How can you access the shadowed class variable x:

- Using: this.x
- Using: super.x
- Using: myClass.x
- Using: class.x
- None of the above.

1.8. Which of the following statements is correct?

- A class must have instance variables.
- A class must have a main method.
- A class can have multiple methods with the same name.
- The Java code of a class is stored in the heap whereas, its variable are stored in the System stack.
- A class can not have more than one constructor.
1.9. Suppose we initialize: \texttt{int [ ] [ ] a = new int[3][4] ;}
What is the type of \texttt{a[0]}?
- \texttt{int}
- \texttt{int[ ]}
- \texttt{int[3]}
- \texttt{int[4]}
- \texttt{int[ ][ ]}

1.10. Consider the following code fragment:

```java
public static void mystery(int[ ] a, int m)
{
    int n = a.length;
    for(int i=0; i<n; i++)
    {
        int tmp = a[i];
        int j = (i+m) \% n;
        a[i] = a[j];
        a[j] = tmp;
    }
}
```

What is the value of the array \texttt{a=\{1,2,3,4\} after calling mystery(a,1) }?

- \texttt{\{1, 2, 3, 4\}}
- \texttt{\{1, 3, 4, 2\}}
- \texttt{\{2, 3, 4, 1\}} (1 credit for this answer - misses last exchange)
- \texttt{\{3, 4, 2, 1\}}
- \texttt{\{2, 1, 4, 3\}}
Question 2. (10 pts) Package Access

Question 2(a): Consider classes ClassA and Question2a below. For each commented statement in class Question2a, indicate whether it causes a compile error.

```java
package myPackage;

public class ClassA
{
    protected int a;
    int b;
    public int c;
}
```

In some other file:

```java
package someOtherPackage;

public class Question2a
{
    public static void main( String[] args )
    {
        ClassA x = new ClassA();
        x.a = 1; // Error (not a derived class and not in same package)
        x.b = 1; // Error (not a derived class and not in same package)
        x.c = 1; // No error (public) (1 pt)
    }
}
```

Question 2(b): Consider class Question2b below (with the same ClassA as above). For each commented statement, indicate whether it causes a compile error.

```java
package myPackage;

public class Question2b
{
    public static void main( String[] args )
    {
        ClassA x = new ClassA();
        x.a = 1; // No error (Package member can access protected)
        x.b = 1; // No error (Package member can access default)
        x.c = 1; // No error (public) (1 pt)
    }
}
```
Question 3. (20 pts, 2 per blank) Fill in the Blank

Put an appropriate phrase or expression in each blank.

In order to avoid an infinite recursion, a recursive function should always include a(n) base case.

In modular programming, we break a complex problem into simpler subproblems, and these subproblems should be independent.

A class may have multiple constructors, as long as they are distinguished by their signature/number or type of the parameters.

If s is a string, a lowercase version of s is s.toLowerCase().

If x and y are strings, we may test whether x should strictly precede y in sorted order, using the boolean expression x.compareTo(y).

If int n is a 3-digit integer (like 123 or 876), then its middle digit is the value of the expression (n/10)%10.

If int[][] a is initialized as rectangular array, then in the expression a[i][j], the maximum legal value of index i is a.length, and the maximum legal value of index j is a[0].length.

If we use binary search to look for 7 in the array {0,1,2,3,4,5,6}, we only examine these elements in the array: 3, 5 and 6.
Question 4. (10 pts) Inheritance

Consider these classes Person and Student:

```java
public class Person {
    private String name;
    private int id;
    public Person(String name, int id) {
        this.name = name;
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public int getId() {
        return id;
    }
}

public class Student extends Person {
    private double gpa;
    public Student(String name, int id, double gpa) {
        super(name, id);
        this.gpa = gpa;
    }
    public double getGPA() {
        return gpa;
    }
}
```

Consider the following program, using classes Person and Student:

```java
public class Test {
    public static void main(String[] args) {
        Person a = new Person( "Mark", 123 ); // Statement A
        Student b = new Student( "Mary", 456, 3.4 ); // Statement B
        String name = b.getName(); // Statement C
        Person c = b; // Statement D
        Student d = a; // Statement E
    }
}
```

For each commented statement in main(), state whether it causes an error (of whatever kind). If it does, explain the reason why it causes an error:

**Statement A**: Error Y / N ? **No error**

**Statement B**: Error Y / N ? **No error**

**Statement C**: Error Y / N ? **No error**

**Statement D**: Error Y / N ? **No error** (Upcasting)

**Statement E**: Error Y / N ? **Error** (Downcasting)
Question 5. (10 pts) Writing (class Die)

Write a class Die that simulate a die with \( N \) sides ("die" is the singular form of "dice"). The sides have face values \( 1, 2, 3, \ldots, N \). Whenever the die is rolled, one of the \( N \) sides comes up, each with equal probability.

Complete the following class definition. Use Math.random(), which returns a double precision floating point number, chosen uniformly in the interval \([0,1)\).

```java
public class Die {
    /*
    * Define any instance variables that you want here:
    */
    private int N;  // 2 pts, 1 pt deduction for private

    /*
    * Complete the constructor "Die( int nSides )"
    * nSides is the number of sides of the die
    */
    public Die( int nSides )
    {
        N = nSides;
    }

    /*
    * Complete the method "int roll( )"
    * Each time roll( ) is invoked, it returns one of the numbers
    * \( 1 \ 2 \ 3 \ \ldots \ N \) (N = the number of sides)
    * with equal probability
    */
    public int roll( )
    {
        return (int) (1 + N*Math.random());
    }
}
```
Question 6. (10 pts) Writing (recursive rangeSum)

Write a static recursive method rangeSum(a, b) that has two integer parameters a and b. If a ≤ b, it should return the sum:

\[ a + (a + 1) + (a + 2) + \ldots + b. \]

Otherwise, it should return zero. For example:

- rangeSum(4, 2) returns 0, because 4 > 2.
- rangeSum(2, 4) returns 9, because 2 + 3 + 4 = 9.

Answer (use recursion, not a loop or a clever formula):

```java
public class Question6 {
    public static int rangeSum(int a, int b) {
        if (a > b) {
            return 0; // base case 1: 2 pt
        } else if (a == b) {
            return a; // base case 2: 2 pt
        } else {
            int sol;
            sol = rangeSum(a, b-1); // or: rangeSum(a+1, b)
            return (b + sol); // or: a + sol
        }
    }
}
```
Question 7. (10 pts) Writing (method countCommon)

Write a static method countCommon(int[] a, int[] b) that takes two integer array parameters a and b, and returns the number of common elements (elements in both arrays). You may assume that there are no repeated elements in a, and no repeated elements in b. For example:

- countCommon({1,4,3,2}, {2,5,1}) returns 2, because 1 and 2 are in both arrays.
- countCommon({2,5}, {1,3,2,4}) returns 1, because only 2 is in both.
- countCommon({2,3,1}, {5,4}) returns 0, because no elements are in both.

In your solution, start with a method isPresent(int[] a, int n) that returns true if the element n is present in array a, false otherwise. Use this method in your countCommon method. Answer:

```java
public class Question7 {
    public static boolean isPresent(int[] a, int n) {
        /* -----------------------------------------------
        Check if n is present in a
        ----------------------------------------------- */
        for (int i=0; i < a.length; i++)
            if (a[i] == n) // present
                return true;

        /* -----------------------------------------------
        We arrive here only when n is not in a
        ----------------------------------------------- */
        return false;
    }

    public static int countCommon(int[] a, int[] b) {
        int count;    // Count the common elements

        /* -----------------------------------------------
        Algorithm:
        for each number in array b[]
            if number is present in a[]
                count++
        ----------------------------------------------- */

        for (int i=0; i < b.length; i++)
            if (isPresent(a, b[i]) // present
                count++;

        return count;
    }
}
```