KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS

Information and Computer Science Department

2011/2012 Second Semester (Term 112)

ICS102: Introduction to Computing (2-3-3)

# FINAL EXAM

Saturday, May 19th 2012, 07:00 PM – 09:00 PM

120 MINUTES

Student Information

| Name: | KEY | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| ID: | | | | | | | | | |

Circle your section

| | | |
|---|---|---|
| Al-Khoraidly | SM 8:00 – 8:50 am | SM 10:00 – 10:50 am |
| Al-Turki | SM 9:00 – 9:50 am | SM 11:00 – 11:50 am |

| Question No. | Maximum Score | Score |
|---|---|---|
| 01 | 20 | |
| 02 | 15 | |
| 03 | 20 | |
| 04 | 20 | |
| 05 | 25 | |
| TOTAL | 100 | |

**Question 1 (20 points):**

Choose the correct answer in the following questions:

1. If the method: `public float aMethod(float a, String b)` is in the class, which of the following is a legal overloading:
   a. `public double aMethod(float x, String y)`
   b. `public float aMethod2(float x, float y)`
   c. `public float aMethod(String a, float b)`
   d. None of the above

2. The error in this code is:
```
public static String half(int c, String d){
   int r = c/2;
   String x = d.substring(r);
}
```
   a. The variable `r` is not in the parameter list
   b. Missing return statement
   c. The two parameters `c` and `d` are separated by a comma not semicolon
   d. `c` and `d` are not defined inside the method

3. Given the following method header:
```
public static int meth(int a, double b)
```
which of the following is a legal call?
   a. `int x = meth(2.0, 5.5);`
   b. `int [] x = meth(0, 3);`
   c. `double x = meth(3, 3);`
   d. `boolean x = meth(1, 2.5);`

4. Given the following code:
```
int [][] a = {{1, 2}, {3, 4, 5}};
a[1] = a[0];
```
which of the following is true?
   a. `a` is a ragged 2D array
   b. Second row has only 2 elements
   c. `a[1][0] == a[0][1]`
   d. None of the above

5. For this code is:
```
char [][] a = {{'a', 'b', 'c'}, {'d'}, {'e', 'f'}};
a[1] = 'g';
```
which of these statements is true?
   a. The element `a[2][1]` is out of bounds
   b. `a.length` and `a[2].length` are equal
   c. Incompatible assignment for `a[1]`
   d. None of the above

6. Which of the following is true about constructors?
   a. The no-argument constructor is always provided by default to the class
   b. Copy constructors can be overloaded
   c. The rules for deciding which method to be used when calling a constructor to create an object are the same as those for calling methods
   d. Both (a) and (c) are true

7. Which of the following code adds the elements at odd indices of the array `ages`?

   a.
   ```
   for(int i = 0; i < ages.length; i++)
       sum += ages[i];
   ```
   b.
   ```
   for(int i = 1; i < ages.length; i++)
       sum += ages[i];
   ```
   c.
   ```
   for(int i = 1; i < ages.length; i += 2)
       sum += ages[i];
   ```
   d.
   ```
   for(int i = 1; i < ages.length; i +=3)
       sum += ages[i];
   ```

8. The error in this code is:
   ```
   public static void f(double a){
      if(a < 0) return;
      System.out.println(Math.sqrt(a));
   }
   ```
   a. The return statement should not be in a void method
   b. The print statement will be executed even if `a` is negative because there is no else
   c. There is no error in the method
   d. The variable `a` is not initialized

9. The error in this code is:
   ```
   public static boolean m(){
      double x = Math.log(Math.E);
      return x;
   }
   ```
   a. The method does not have parameters
   b. The return value is not of the correct type
   c. There is no error in the method
   d. The name of the method is illegal

10. Given the class:
    ```
    class A{
       public static void main(String [] args){
          A o = new A();
          o.printResult(1, "abc", 4);
       }
       private void printResult(long a, String b, int c){
          System.out.println(a + b + c);
       }
    }
    ```
    Which of the following statements is true?
    a. The code will not compile because the `printResult` method is nonstatic and cannot be called from the static method `main`
    b. The code will compile and run and will display `1abc4`
    c. The code will not compile because there is no constructor in the class
    d. The code will not compile because the `printResult` method is private

**Question 2 (15 points):**

Find the output of the following Java code snippets:

**I)**

```
int [ ][ ] x ={{5, 6, 8}, {2, 4, 3},{7, 1, 9}};
int temp;
for(int k = 1; k < 3; k++){
  temp = x[3 - k][0];
  x[3 - k][0] = x[0][k];
  x[0][k] = temp;
}
for(int i = 0; i < x.length; i++) {
  for(int j = 0; j < x[i].length; j++)
    System.out.print(x[i][j] + "  ");
  System.out.println();
}
```

**OUTPUT**

```
5  7  2
8  4  3
6  1  9
```

**II)**

```
int [] arr = {75, 57, 10, 81, 63, 32, 24};
int min = 0;
for(int i = 1; i < arr.length; i++)
  if(arr[i] < arr[min])
     min = i;
for(int i = 0; i < arr.length; i++)
   arr[i] = arr[i] - arr[min];
System.out.print(arr[0] + "  " + arr[3] + "  " + arr[6]);
```

**OUTPUT**

```
65  81  24
```

**III)**

```
public static void main(String []args){
  int [] a = {1, 2, 3, 4};
  increment(a[1]);
  increment(a);
  System.out.println(a[1]+"  "+a[a[1]]);
}
public static void increment(int b){
  b += 2;
  System.out.println(b);
}
public static void increment(int [] b){
  b[1] += b[0];
}
```

**OUTPUT**

```
4
3  4
```

**IV)**

```
public class Test {
  public static void main(String [] args) {
    MyClassA a = new MyClassA();
    MyClassB b1 = a.getB();
    MyClassB b2 = b1;
    b2.setX(20);
    System.out.println(a.getX()+ "  " + b1.getX());
    System.out.println(a.getY()+ "  " + b2.getX());
  }
}
class MyClassA {
  int y = 10;
  MyClassB b = new MyClassB();
  public int getY() { return y; }
  public int getX() { return b.getX(); }
  public MyClassB getB() { return b; }
}
class MyClassB {
  int x = 10;
  public int getX() { return x; }
  public void setX(int x) { this.x = x; }
}
```

**OUTPUT**

```
20  20
10  20
```

**Question 3: (20 points)**

Write a program that reads the lines of text in a text file `input.txt` and prints them to the screen in the reverse order (i.e., the last line is printed first to the screen), such that all letters are capitalized (uppercased). Assume that the file has at most 50 lines. Hint: read the lines to an array of Strings.

Example:

If the file contains the following:

```
Hello guys..
It is Saturday..
Sunny outside..
Bye..
```

The output on the screen should look like:

```
BYE..
SUNNY OUTSIDE..
IT IS SATURDAY..
HELLO GUYS..
```

---

```
import java.util.Scanner;
import java.io.FileInputStream;
import java.io.FileNotFoundException;

class FileProcessor {
 public static void main(String[] args) throws FileNotFoundException {
   Scanner input = new Scanner(new FileInputStream("input.txt"));
   String [] lines = new String[50];
   int i = 0;
   while(input.hasNext()) {
     lines[i] = input.nextLine();
     i++;
   }
   for(int j = i - 1; j >= 0; j--)
      System.out.println(lines[j].toUpperCase());
   input.close();
  }
}
```

---

---

---

---

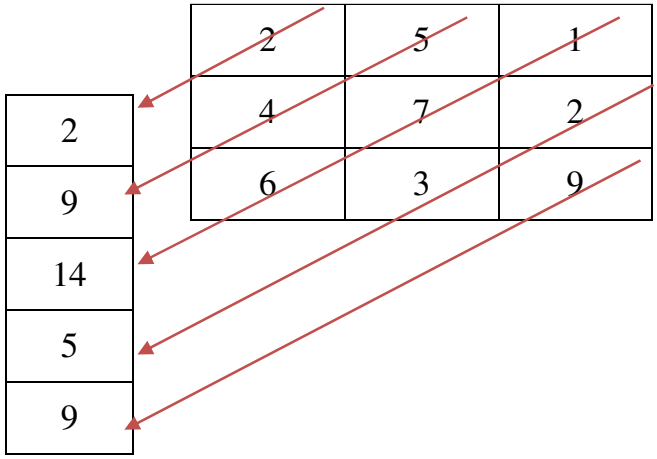
---

---

**Question 4: (20 points)**

Write a method that takes as input a square 2D array (same number of rows and columns) of doubles and computes the sum of elements along each of the counter-diagonals (anti-diagonals).

Example:




```
public double [] antiDiagonalSum (double [][] a)
{
  // assuming a square array
  double [] sums = new double [2*a.length-1];

  for(int i=0; i<a.length; i++)
    for(int j=0; j<a.length; j++)
      sums[i+j] += a[i][j];

  return sums;
}
```

**Question 5: (25 points)**

**A.** Suppose we want to implement a class, called "Airplane", that defines a data type for airplanes. Complete the implementation of this class "Airplane" as instructed by the enclosed comments below.

```
class Airplane {

 /* A constant 'MAX' that has the value 200
  */

static final int MAX = 200;

 /* Three instance variables:
  * 1) 'id': A number identifying the airplane
  * 2) 'model': A string specifying the airplane model
  * 3) 'seats': A Boolean array. The meaning is that the i'th element in
  *             the array has the value true if and only if the i'th
  *             seat is available (not reserved).
  */

private int id;
private String model;
private boolean [] seats;

  /* A three-argument constructor to initialize all the instance
   * variables to values given as parameters
   */

public Airplane( int id, String model, boolean [] seats)
{
  this.id = id;
  this.model = model;
  this.seats = seats;
}

  /* A two-argument constructor to initialize the 'id' and 'model' to
   * values given as parameters, and the 'seats' to an array of size MAX
   * where all seats are available
   */

public Airplane( int id, String model)
{
  this.id = id;
  this.model = model;
  seats = new boolean [MAX];

  for(int i=0; i<seats.length; i++)
    seats[i] = true;
}
```

```
  /* A copy constructor
   */

public Airplane( Airplane a )
{
  id = a.id;
  model = a.model;
  seats = a.seats.clone();
}

 /* An accessor method for 'id'
  */

public int getId( )
{
  return id;
}

 /* A mutator method for 'model'
  */

public void setModel(String model)
{
  this.model = model;
}

 /* A method 'getFirstAvailableSeat' that returns the index of the first
  * seat available in 'seats'
  */

public int getFirstAvailableSeat()
{
  for(int i=0; i<seats.length; i++)
    if(seats[i])
      return i;

  // no available seats
  return -1;
}

/* A method 'reserveSeat' that attempts to reserve the i'th seat, where
 * i is given as a parameter (which must be a valid index in the array
 * of seats). The method returns true if the reservation attempt is
 * successful, and false otherwise.
 */

public boolean reserveSeat(int i) {
  if(i >= 0 && i < seats.length && seats[i])
  {
    seats[i]=false;
    return true;
  }
  return false;
}
```

```
/* A method 'isFull' that returns true if there is no available seat,
 * and false otherwise.
 */

public boolean isFull() {
  for(int i=0; i<seats.length; i++)
    if(seats[i])
      return false;

  return true;
}

/* A 'toString' method that returns a string containing the airplane's
 * id and model only
 */

public String toString() {
  return "ID: " + id + ", Model: " + model;
}

/* An 'equals' method that returns true if and only if this airplane has
 * the same id and model as the other airplane given as a parameter
 */

public boolean equals(Airplane a) {
  return id == a.id && model.equalsIgnoreCase(a.model);
}
```

**B.** To test the class above, complete the implementation of the following simple class "AirplaneDemo" as instructed by the enclosed comments below.

```
public class AirplaneDemo {
    public static void main(String [] args) {

     /* Create two instances 'a1' and 'a2' of the class Airplane:
      * a1 has id: 710, and model: "Boeing 747"
      * a2 has id: 720, and model: "Airbus A320"
      */

     Airplane a1 = new Airplane( 710, "Boeing 747");

     Airplane a2 = new Airplane( 720, "Airbus A320");

     /* Reserve seat 22 in a1, and then print out whether this
      * reservation was successful
      */

     if(a1.reserveSeat(22))
         System.out.println("Reservation was successful");
     else
         System.out.println("Reservation was not successful");
```

```
        /* Print out the first available seat in a1
         */

        System.out.println(a1.getFirstAvailableSeat());

        /* Print out whether the two airplanes a1 and a2 are the same
         */

        if(a1.equals(a2))
            System.out.println("Same airplane.");
        else
            System.out.println("Different airplanes.");


        /* Print out the two airplanes a1 and a2
         */

        System.out.println("First airplane:\n" + a1);
        System.out.println("Second airplane:\n" + a2);

    }
}
```