

Computer Sciences 302
Final Exam, 20%

Monday 12/20, 2010

Print last name: _____, first: _____

Signature: _____ CS login: _____

Circle Your Lecture	**Lec 1** Skrentny	**Lec 2** Skrentny	**Lec 3** Dalibor	**Lec 4** Dalibor	**Lec 5** Finn	**Lec 7** Dan	**Lec 8** Finn	**Lec 9** Alex	**Lec 10** Alex

This exam is composed of two parts. *Part I is to be answered by filling in your choice using a #2 pencil on the separate answer sheet. Part II is to be answered by writing your answers in this examination booklet.*

Before you Begin:

(1) Take a separate answer sheet and write your UW student ID number on it.
(2) Turn in your UW student ID.
(3) **On the separate answer sheet:**
- Fill in the bubbles corresponding to each digit of your UW student ID number.
- Write your name and then fill in the bubbles corresponding to each letter.
- In the "Special Codes" section under letter "A" write your lecture number and fill in the corresponding bubble, and under letter "B" write P but do not fill in a bubble.

(4) **On this examination booklet:**
- Print and sign your name above.
- Write your CS login and circle your lecture above.

(5) Check that there is a total of 14 pages in this exam.
(6) You may not use notes, books, calculators (or any other electronic devices), or neighbors on this exam. Turn off and put away your cell phone, pager, pda, etc. now.
(7) **You have 2 hours to complete the exam**. Use your time wisely.
(8) We can't provide hints but if you need an exam question clarified or feel there is an error, please bring this to our attention. If needed, **corrections will be written on the board**.

When you've Finished:

(9) For Part I, double check that you have correctly marked the bubbles on your answer sheet. Only marks on the answer sheet matter. Marks in this examination booklet don't count.
(10) Turn in this examination booklet and your answer sheet, and make sure we return your ID.

Parts	Number of Questions	Question Format	Possible Points	Score
I	21 (1 bonus question)	Multiple Choice	63 (3 bonus points)	
II	3	Written Answers	21	
		Total	81	

Exam Reference Page

Methods from the `java.lang.String` class: (*REMEMBER `Strings` use 0-based indexing)

```
String(String str)      // create a String object given another String
int length()            // returns # of characters in this string
char charAt(int index)  // returns the character at the zero-based index
boolean equals(String s)// returns true if the contents of the String
                        // instance is the same as that of the string s
String toUpperCase()    // returns a new String that is all of the chars in this
                        // String converted to upper case
*Note: See the Comparable interface below for the compareTo method.
```

Methods from the `java.io.PrintWriter` class:

```
PrintWriter(String filename) throws FileNotFoundException
                        // Creates a PrintWriter for the given file name
PrintWriter(Filewriter out)// Creates a PrintWriter from the given FileWriter
void close()            // Closes the stream and any associated file
void print(String s)    // Prints given string
void println(String s)  // Prints given string followed by a newline
```

Methods from the `java.io.File` class:

```
File(String filename) throws FileNotFoundException
                        // Creates a File for the given file name
void close() throws IOException // Closes the stream and any associated file
```

Methods from the `java.util.ArrayList` class (`T` is the type of element):

```
ArrayList<T>()          // Creates an empty array list to store T
boolean add(<T> m)      // Appends the specified T to the end of the list
                        // and returns true iff the list is changed
void add(int index, <T> m)//Inserts the specified T at the specified index
<T> get(int index)      // Returns the T at the specified index
int size()              // Returns the number of elements in this list
```

Methods from the `java.util.Random` class:

```
Random()                // Creates a new random number generator.
int nextInt()           // Returns the next pseudorandom int value.
int nextInt(int n)      // Returns the next pseudorandom int value
                        // between 0 (inclusive) and n (exclusive).
```

Methods from the `java.util.Scanner` class:

```
Scanner(System.in)      // Creates a Scanner object that reads from the keyboard
Scanner(FileReader in)  // Creates a Scanner object that reads from FileReader
void close()            // Closes the scanner and any associated file
*Note: Tokens are sequences of characters separated by whitespace.
boolean hasNext()       // Returns true iff another token exists in input
String next()           // Returns the next token of input
boolean hasNextInt()    // Returns true iff another int exists in input
boolean hasNextLine()   // Returns true iff another line exists in input
String nextLine()       // Returns the next line of input
*Note: Methods below throw an InputMismatchException if the input type is wrong.
int    nextInt()        // Scans the next token of the input as an int
double nextDouble()     // Scans the next token of the input as a double
```

The `Comparable<T>` interface (`T` is the type being compared):

```
int compareTo(<T> t)    // Returns a negative if this object is less than t,
                        // 0 if they are equal, and a positive if this object
                        // is greater than t
```


Part I Multiple Choice *[21 questions, 3 points each, 63 total points]*

There are 21 question each worth 3 points with a maximum score of 63 points **(there is 1 bonus question)**.
For the questions on the following pages, **choose the one best answer after reading all of the choices**.
Use a #2 pencil to fill in the bubble on your answer sheet that corresponds to your answer for each question.

1.) Consider the following code:

```
for (int i = 1; i != 20; i+=2) {
   //loop body
}
```

How many times will the body of this loop repeat?

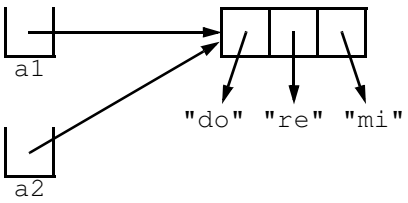
A. 0
B. 1
C. 10
D. 11
E. more than 20

2.) Consider the following code:

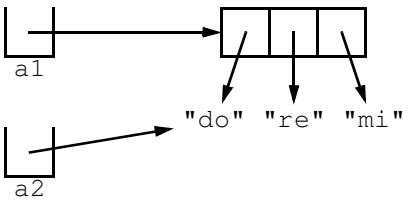
```
String[] a1 = {"do", "re", "mi"};
String[] a2 = new String[3];
for (int i = 0; i < a1.length; i++) {
   a2[i] = new String(a1[i]); //review reference sheet if needed
}
```

Which one of the following is the correct memory diagram after the lines of code above execute?

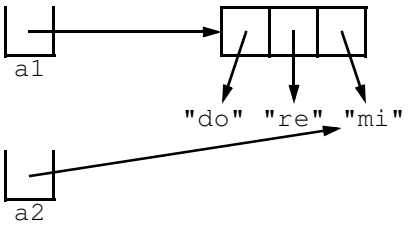
A.



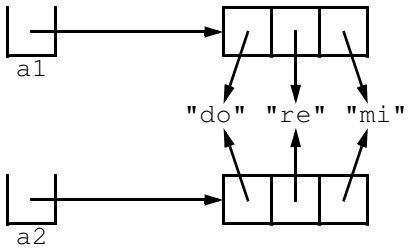
D.



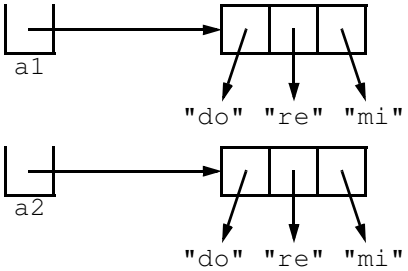
B.



E.



C.



3.) Which one of the following pairs of conditions does ***not*** give the same result under all circumstances (i.e., they are ***not*** equivalent to each other)? Assume `n` is an integer variable and `flag` is a boolean variable.

	Condition 1:	Condition 2:
A.	`!(n <= 22)`	`n > 22`
B.	`flag == true`	`!flag`
C.	`n > 11 || n < 22`	`true`
D.	`n < 22 || n > 22`	`n != 22`
E.	`n == 11 && n == 22`	`false`

4.) Consider the following code fragment, where `i` is an integer variable and `d` is a double:

```
switch (i) {
   case 1:  d = 2.2; break;
   case 2:
   case 3:  d = 7.7; break;
   default: d = 1.1; break;
}
```

This code fragment is equivalent to which one of the following?

A.
```
d = 1.1;
```

B.
```
if (i == 1) {
    d = 2.2;
} else {
   d = 7.7;
}
d = 1.1;
```

C.
```
if (i == 1) {
    d = 2.2;
} else if ((i == 2) || (i == 3)) {
   d = 7.7;
} else {
   d = 1.1;
}
```

D.
```
if (i == 1) {
    d = 2.2;
}
if ((i == 2) || (i == 3)) {
   d = 7.7;
} else {
   d = 1.1;
}
```

E.
```
if (i == 1) {
    d = 2.2;
}
if ((i == 2) || (i == 3)) {
   d = 7.7;
}
if (i > 3) {
   d = 1.1;
}
```

The questions on this page are based on the following code that is *partially* shown below:

```
public class Exceptions {

   public static void main(String[] args){
      try {
         methodX();
         //LINE A - see questions below
         methodY();
      } catch (NullPointerException e) {
         System.out.print("Error 1,");
      } catch (IndexOutOfBoundsException e) {
         System.out.print("Error 2,");
      }
      System.out.print("main done,");
   }

   public static void methodX() {
      try {
         //LINE B - see questions below
      } catch (NullPointerException e) {
         System.out.print("Error 3,");
      }
      System.out.print("methodX done,");
   }

   public static void methodY() {
      try {
         //LINE C - see questions below
      } catch (ArithmeticException e) {
         System.out.print("Error 4,");
      }
      System.out.print("methodY done,");
   }
}
```

5.) What is output if LINE A is replaced with code that will throw a `NullPointerException`?

A. `Error 1,`
B. `Error 1,main done,`
C. `methodX done,methodY done,Error 1,main done,`
D. `methodX done,Error 1,main done,`
E. The program crashes and message about the exception is displayed.

6.) What is output if LINE B is replaced with code that will throw an `IndexOutOfBoundsException`?

A. `Error 2,`
B. `Error 2,main done,`
C. `methodX done,Error 2,main done,`
D. `Error 2,methodY done,main done,`
E. The program crashes and message about the exception is displayed.

7.) If LINE C is replaced with code that will throw a `FileNotFoundException` which methods would require a `throws FileNotFoundException` clause be added to their method header?

A. only `main`
B. only `methodX`
C. only `methodY`
D. only `main` and `methodY`
E. none of the methods since a `FileNotFoundException` is unchecked

8.) Consider implementing a main method that uses command-line arguments to specify a file name. If two command-line arguments are provided, then the first is expected to be either "i" or "o". If it is "i", the second argument specifies an *input* file name. If it is "o", the second specifies an *output* file name.

```
public void main(String[] args) {
   String input, output;
   //...
   if (CONDITION) {
      switch (EXPRESSION1) {
      case 'i': input  = EXPRESSION2; break;
      case 'o': output = EXPRESSION2; break;
      }
   }
   //...
```

Which one of the following replacements for CONDITION, EXPRESSION1 and EXPRESSION2 (appearing twice), when used to complete the method above, processes the command-line arguments as specified?

	CONDITION	EXPRESSION1	EXPRESSION2
A.	`args.length() == 2`	`args[0]`	`args[1]`
B.	`args.length == 2`	`args[1]`	`args[2]`
C.	`args.length == 2`	`args[0].charAt(0)`	`args[1]`
D.	`args.length == 2`	`args[1].charAt(0)`	`args[2]`
E.	`args.length() == 2`	`args[1].charAt(1)`	`args[2]`

9.) Consider the following complete implementation of the `Data` class:

```
public class Data {
   private static int sdata = 0;
   private int idata;
   public  Data() {
      sdata++;
      idata = sdata;
   }
   public void set (int data) {
      idata = data;
   }
   public String toString() {
      return "i:" + idata;
   }
}
```

Consider the following application class:

```
public class DataApp {
   public static void main(String[] args) {
      int data = 11;
      Data d1 = new Data();
      Data d2 = d1;
      d2.set(data);
      d2 = new Data();
      System.out.println(d1 + ", " + d2);
   }
}
```

What is printed when the `DataApp` program is executed?

A. `i:1, i:3`
B. `i:2, i:2`
C. `i:1, i:11`
D. `i:11, i:1`
E. `i:11, i:2`

10.) Consider the following complete implementation of the `Question` class:

```
public class Question {
   public  final double A = 11.22;
   public  double b;
   private double c;
   private void one (double d) {
      this.b = d;
   }
   public double two () {
      return this.c;
   }
   public void three (double e) {
      this.c = e;
   }
}
```

Assume `q` is a properly constructed `Question` object, and it and the statements below are in the `main` method of an `Exam` class. Which one of the following statements will ***not*** result in a compile-time error?

A. `q.A = 1.1;`
B. `q.b = 1.1;`
C. `q.one(q.A);`
D. `q.A = q.two();`
E. `q.three(q.c);`

11.) Consider designing a method that gets x and y coordinates (two integers) from the user as partially shown:

```
public ??? getCoordinates() {
   System.out.print("Please enter the coordinates separated by a space: ");
   int x = in.nextInt();
   int y = in.nextInt();
   //...
}
```

Which of the following techniques could be used to return both integers from this method?

i. In the `getCoordinates` method, use two return statements: `return x;` followed by `return y;`
ii. In the `getCoordinates` method, make an array of integers having two elements. Store the values of x and y in that array, and return it from the method.
iii. Implement a new instantiable class having two integer instance members and a constructor, mutators, and accessors. In the `getCoordinates` method, make an object of this class constructed from the values of x and y, and return this object from the method.

A. none
B. *i* only
C. *ii* only
D. *ii* and *iii* only
E. *i*, *ii*, and *iii*

12.) Which one of the following lists of values would most thoroughly test a program's user input to make sure that it is limited to be in the range from 1 to 11 inclusive?

A. 1, 11
B. 10, 11, 12
C. 0, 1, 2, 3, 9, 10, 11
D. 0, 1, 2, 6, 10, 11, 12
E. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

13.) Consider the following method:

```
public int processString(String s, char c) {
   int n = -1;
   for (int i = 0; i < s.length(); i++) {
      if (s.charAt(i) == c) {
         n = i;
      }
   }
   return n;
}
```

Which one of the following best describes what method `processString` does?

A. It always returns -1.
B. It returns the number of times character `c` is found in `String s`.
C. It returns the number of characters in `String s` that are not character `c`.
D. It returns the position of the *first* occurrence of character `c` in `String s` or -1 if `c` is not found.
E. It returns the position of the *last* occurrence of character `c` in `String s` or -1 if `c` is not found.

14.) Consider the following method:

```
public static void patternFill(char[][] board) {
   boolean x = true;
   for (int r = 0; r < board.length; r++) {
      for (int c = 0; c < board[r].length; c++) {
         if (x) {
            board[r][c] = 'X';
            x = false;
         }
         else {
            board[r][c] = 'O';
            x = true;
         }
      }
   }
}
```

Which one of the following correctly shows the pattern in `board` after it has been filled by this method?

A. A 4 by 4 board would be filled with this pattern:

X	O	X	O
O	X	O	X
X	O	X	O
O	X	O	X

B. A 3 by 4 board be filled with this pattern:

X	O	X	O
O	X	O	X
X	O	X	O

C. A 3 by 3 board would be filled with this pattern:

X	O	X
O	X	O
X	O	X

D. A 4 by 4 board would be filled with this pattern:

O	X	O	X
X	O	X	O
O	X	O	X
X	O	X	O

E. A 3 by 4 board would be filled with this pattern:

O	X	O	X
X	O	X	O
O	X	O	X

15.) Consider the following code fragment where `a` is a 2-dimensional array of integers and `x` is some integer:

```
//location A
for (int r = 0; r < a.length; r++) {
   //location B
   for (int c = 0; c < a[r].length; c++){
      //location C
      if (a[r][c] == x) {
         count++;
      }
   }
   System.out.println(count);
   count = 0;
}
```

We must add `int count = 0;` to complete this code fragment. At which of the locations labeled above can this code be added so that it ***doesn't*** cause a compile-time error?

A. only location A
B. only location B
C. only location C
D. only locations A and B
E. only locations B and C

16.) Which one of the following statements about Java data types is *false*?

A. Data types are classified into two groups: primitives and references.
B. Reference variables can be aliases but primitive variables cannot.
C. Java allows arrays of primitive data types and arrays of reference data types.
D. A primitive value is copied whenever it is passed as a parameter or returned from a method.
E. A software object is copied whenever its reference is passed as a parameter or returned from a method.

17.) Consider the following method that is passed an array of integers that are in the range of 0 to 9 inclusive:

```
public static boolean doSomething (int[] d) {

   boolean[] b = new boolean[10]; //elements are initialized to false

   for (int i = 0; i < d.length; i++) {
      if (b[d[i]]) {
         return false;
      }
      else {
         b[d[i]] = true;
      }
   }
   return true;
}
```

Which one of the following *best* describes what method `doSomething` does?

A. It always throws an `ArrayIndexOutOfBoundsException`.
B. It returns true if and only if the array of integers has no duplicates.
C. It returns true if and only if the array of integers has all of the digits from 0 to 9.
D. It returns true if and only if the array of integers is sorted in increasing order.
E. It returns true if and only if the array of integers is sorted in decreasing order.

The questions on this page are based on the following *partially shown* classes that could be used for program 4**:**

```
public class Square {
   private int     gui;    //what to display in the graphical interface
   private int     mcount; //count of surrounding mines
   private boolean mine;   //set to true if this square has a mine

   public Square(int g) {
      this.gui = g; this.mcount = 0; this.mine = false;    }
   public boolean hasMine()            { return this.mine; }
   public void    setMineCount(int m)  { this.mcount = m;  }
}
public class Minefield {
   private Square[][] minefield;

   public Minefield(int rows, int cols){
      //assume this properly constructs a mine field of squares }
}
```

18.) Consider the following incomplete private helper method that's added to the `Minefield` class above:

```
private int getSurroundingMineCount(int row, int col) {
   int count = 0;
   for (int r = row - 1; r <= row + 1; r++) {
      for (int c = col - 1; c <= col + 1; c++) {
         try {
            if (CONDITION) { count++; }
         }
         catch (ArrayIndexOutOfBoundsException e) {
            STATEMENT
   }  }  }  //braces condensed to one line to save space
   return count;
}
```

Which one of the following four replacements for `CONDITION` and `STATEMENT`, when used to complete the method above, will return the count of mines in squares *surrounding* the square at location `row`, `col`?

	CONDITION	STATEMENT
A.	`minefield.hasMine(r, c)`	`//nothing`
B.	`minefield[r][c].hasMine()`	`count--;`
C.	`minefield[r][c].hasMine() && !(r == row && c == col)`	`//nothing`
D.	`minefield[r][c].hasMine() && !(r == row && c == col)`	`count--;`

19.) Assume the `getSurroundingMineCount` method above is correctly implemented. Consider the following incomplete instance method that's added to the `Minefield` class above:

```
public void setAllMineCounts() {
   for (int r = 0; r < minefield.length; r++) {
      for (int c = 0; c < minefield[r].length; c++) {
         STATEMENT
}  }  }  //braces condensed to one line to save space
```

Which one of the following replacements for `STATEMENT`, when used to complete the method above, will set the *surrounding* mine counts for all of the squares in the mine field?

	STATEMENT
A.	`minefield.setMineCount(r, c);`
B.	`minefield[r][c].getSurroundingMineCount(setMineCount(r, c));`
C.	`minefield[r][c].setMineCount(getSurroundingMineCount[r][c]);`
D.	`minefield[r][c].setMineCount(getSurroundingMineCount(r, c));`
E.	`minefield[r][c].getSurroundingMineCount(setMineCount[r][c]);`

20.) Assume a class named `Movie` has a method named `getTitle` that returns a `Movie` object's title. Also assume an `ArrayList` named `database` has had `Movie` objects added to it. Which one of the following correctly determines whether the title of the first movie in the database is equal to "Koyaanisqatsi"?

A. `database.get(0).getTitle() == "Koyaanisqatsi"`
B. `database.get(1).getTitle() == "Koyaanisqatsi"`
C. `database.get(0).getTitle.equals("Koyaanisqatsi")`
D. `database.get(0).getTitle().equals("Koyaanisqatsi")`
E. `database.get(1).getTitle().equals("Koyaanisqatsi")`

21.) Consider the following class:

```
public class Parameters {

   public static void mystery(int[] b, int x) {
      b[0] = 1;
      b = new int[4];
      b[3] = 5;
      x = 2;
   }
   public static void main(String[] args) {
      int[] a = {9, 8, 7, 6};
      int x = 1;
      mystery(a, x);
      a[x] = 3;
      for (int i = 0; i < a.length; i++)
         System.out.print(a[i] + " ");
   }
}
```

Which one shows what the `Parameters` program displays when executed?

A. `1 3 7 5`
B. `1 3 7 6`
C. `1 8 3 5`
D. `1 8 3 6`
E. The program displays a message about a `ArrayIndexOutOfBoundsException`.

Part II Written Answers *[3 question, 21 total points]*

Write your answers to the written question in this examination booklet. If you need more room use the back of the last page and indicate on this page where your answer continues.

1.) [7 points] Complete the method below that is passed two `Scanner`s that are already properly opened to two text files. This method creates an output file that interleaves the lines of the two input files. For example, if `file1` and `file2` are opened to the files shown below, then the following would be in the output file after the method finishes:

Input File 1:
```
Well then who's on first?
I mean the fellow's name.
```

Input File 2:
```
Yes.
Who.
```

Resulting Output File:
```
Well then who's on first?
Yes.
I mean the fellow's name.
Who.
```

Name the output file "output.txt" and return false if there is a problem opening that file, otherwise return true. You may assume that any necessary import statements have already been coded and both input files have the same number of lines greater than or equal to 0.

```
public boolean interleave (Scanner file1, Scanner file2) {
```

2.)

[7 points] Assume that a `Time` class has been implemented to represent times of the day with hours in the range of 0 - 23 and minutes in the range 0 - 59. This class has a constructor that is passed two integers, the first for the hours and the second for the minutes. If the values passed to the constructor are out of range the class throws an `IllegalArgumentException`. **Write a code fragment that uses exception handling** to contruct a valid `Time` object from the user's input. If input that's out of range is entered, display the exception's error message (hint: use `getMessage()`) and enable the user to re-input. If an `InputMismatchException` is thrown due to input of the incorrect type, display "Invalid Input" and enable the user to re-input. You may assume that any necessary import statements have already been coded.

```
Scanner in = new Scanner(System.in);
```

3.) [7 points] Modify the `Fraction` class, partially shown below, so that it implements the `Comparable` interface. This class represents a fraction, such as 1/4 or 2/3, where the first integer (e.g., 1, 2) is the numerator and the second (e.g., 4, 3) is the denominator. Hint: Convert the fractions to floating-point values.

```
public class Fraction {

   private int numerator;
   private int denominator;

   public Fraction(int n, int d) {
      this.numerator = n;
      this.denominator = d;
   }

   public int getNumerator() {
      return this.numerator;
   }

   public int getDenominator() {
      return this.denominator;
   }
```