CS-1331-O1 Exam 3

Vidit Dharmendra Pokharna

TOTAL POINTS

82 / 101

QUESTION 1

1 True/False 7.5 / 10

+ 2.5 pts 1: F

✓ **+ 2.5 pts** *2: F*

✓ **+ 2.5 pts** *3: F*

✓ **+ 2.5 pts** *4: T*

+ 0 pts Incorrect

QUESTION 2

2 Multiple Choice 6 / 10

✓ **+ 3 pts** *5: Last among the list of catch blocks for a given try block*

✓ **+ 3 pts** *6: Call stack*

+ 4 pts 7: Java doesn't enforce the handling of exceptions extending from RuntimeException

+ 0 pts Incorrect

QUESTION 3

3 Q8 4 / 4

✓ **- 0 pts** *6: 1, 3: 2, 5: 3*

- 2 pts At most one incorrect, missing, or extra value

- 4 pts Incorrect

QUESTION 4

4 Q9 5 / 5

✓ **- 0 pts** *Correct 2 3 8 1 6*

- 5 pts Incorrect

QUESTION 5

5 Q10 4 / 4

✓ **- 0 pts** *Correct*

- 1 pts Merge Sort Big-O incorrect, should be O(n log n)

- 1 pts Merge Sort Growth Rate incorrect, should be Linearithmic

- 1 pts Selection Sort Big-O incorrect, should be O(n^2)

- 1 pts Selection Sort Growth Rate incorrect, should be Quadratic

- 0.5 pts Any of the Big O notations doesn't contain the "O()" part but is otherwise correct

- 1 pts Mixed up Big-O notation and Growth Rate

- 4 pts Incorrect

- 0.5 pts ****

QUESTION 6

6 Q11 4 / 4

✓ **- 0 pts** *Correct*

- 2 pts Doesn't mentions Merge Sort (or Selection Sort for an incorrect response to Q10, the response should be the one with smaller growth rate or Big-O)

- 2 pts Doesn't justify algorithm based on comparison of Big-O or growth rate

QUESTION 7

7 Q12 3 / 3

✓ **- 0 pts** *Correct*

- 0.5 pts Declared Type v1 incorrect, should be A

- 0.5 pts Object Type v1 incorrect, should be B

- 0.5 pts Declared Type v2 incorrect, should be B

- 0.5 pts Object Type v2 incorrect, should be B

- 0.5 pts Declared Type v3 incorrect, should be C

- 0.5 pts Object Type v3 incorrect, should be A

- 3 pts Incorrect

QUESTION 8

8 Q13 13.5 / 18

+ 0 pts Incorrect

A

✓ **+ 1.5 pts** *Indicates correct output: 1*

✓ **+ 1.5 pts** *Indicates there is no run time or compiler error by giving an output*

B

+ 1.5 pts Indicates compiler error

+ 1.5 pts Correct reasoning: stringA is not a method of Object (or similar)

C

+ 1.5 pts Indicates correct output: I am B

✓ **+ 1.5 pts** *Indicates that there is no run time or compiler error by giving an output*

D

✓ **+ 1.5 pts** *Indicates run time error or unchecked exception*

✓ **+ 1.5 pts** *Mentions ClassCastException*

E

✓ **+ 1.5 pts** *Indicates correct output: 2*

✓ **+ 1.5 pts** *Indicates that there is no run time or compiler error by giving an output*

F

✓ **+ 1.5 pts** *Indicates compiler error*

✓ **+ 1.5 pts** *Correct reasoning: A cannot be converted into B / value from instance of A cannot be stored in a variable with declared type B (or similar)*

- 1 pts One time deduction for printing extra output (quotations or extra text)

QUESTION 9

9 Q14 9 / 12

A

- 0 pts AG-200

✓ **- 1 pts** *One more, less, or different from the composing elements: A G -200 (example: BG-200 or G-200) OR correct elements in incorrect order (example: GA-200)*

- 2.5 pts Two more, less, or different from the composing elements: A G -200 (example: -200 or ABCG-200) OR one element different and incorrect order (example: -200G)

- 4 pts Incorrect

B

- 0 pts BF

- 1 pts FB

✓ **- 2 pts** *One more, less, or different from the composing elements: B F (example: B, BFG, AF)*

- 4 pts Incorrect

C

✓ **- 0 pts** *DE*

- 1 pts ED

- 2 pts One more, less, or different from the composing elements: D E (example: D, DEG, DF)

- 4 pts Incorrect

- 0.5 pts One time deduction: any answer with awarded points has elements separated by space/commas or newline or has printed with quotes

- 12 pts All Incorrect / Missing

QUESTION 10

10 Q15 **8 / 8**

+ 0 pts Incorrect

A

✓ **+ 2 pts** *Comparable*

+ 1 pts Comparable[]

+ 1 pts Comparable$$<T$$>

✓ **+ 2 pts** *B: Comparable*

C

✓ **+ 2 pts** *list[currIndex].compareTo(list[mindex]) OR list[mindex].compareTo(list[currIndex])*

✓ **+ 2 pts** *< 0 if "list[currIndex].compareTo(list[mindex])", > 0 if "list[mindex].compareTo(list[currIndex])"*

- 1 pts Minor syntax error

QUESTION 11

11 Q16 **3 / 3**

✓ **+ 0.75 pts** *public interface*

+ 0.5 pts interface (not public)

✓ **+ 0.75 pts** *ExamInterface*

✓ **+ 1 pts** *void process(String s) (may be prefixed with public, abstract, or both, although discouraged)*

✓ **+ 0.5 pts** *Method ends with semicolon instead of {}*

- 1 pts Minor syntax error

+ 0 pts Incorrect

QUESTION 12

12 Q17 **3 / 3**

✓ **+ 0.5 pts** *public class Turkey*

✓ **+ 0.5 pts** *extends Bird*

✓ **+ 0.75 pts** *implements*

Comparable in header

✓ **+ 1.25 pts** *Comparable<Turkey\>*

+ 0.5 pts Comparable

+ 0 pts Incorrect

- 1 pts Minor syntax error (other than adding { at the end)

QUESTION 13

13 Q18 **4 / 4**

+ 0 pts Incorrect

✓ **+ 1 pts** *public int compareTo*

✓ **+ 1 pts** *Parameter is of type Turkey*

Method Body

✓ **+ 2 pts** *return wattleSize - t.wattleSize; OR equivalent code*

+ 1 pts Returns flipped value (can return correct value in some cases) but is otherwise correct

+ 1 pts Returns positive and negative values but not zero when equal

- 0.5 pts Uses getters

- 0.5 pts Minor syntax error

QUESTION 14

14 Q19 **7 / 12**

- 0 pts Correct (sample solution):

```
import java.io.\*;
import java.util.\*;
```

```
public class OddToAnother {
public static void main(String[] args) throws
IOException {
File in = new File(args[0]);
File out = new File(args[1]);
int lineNumber = 1;
Scanner sc = new Scanner(in);
PrintWriter pw = new PrintWriter(out);
while (sc.hasNextLine()) {
String line = sc.nextLine();
if (lineNumber % 2 == 1) {
pw.println(line);
}
lineNumber++;
}
sc.close();
pw.close();
}
}
```

- 1 pts Missing or incorrect imports (needs to import java.io.File, java.io.IOException, java.io.PrintWriter, java.util.Scanner, imports can use wildcard or be individual)

- 0.5 pts Incorrect class header

- 0.5 pts Incorrect main method header, excluding the "throws IOException" part

- 1 pts main method header doesn't have "throws" in the header

- 0.5 pts main method header has "throws" in the header but the exception thrown isn't IOException (and IOException only)

- 1 pts Doesn't create File object for input file

- 1 pts Doesn't create File object for output file

- 0.5 pts Either the input or output File object instantiations receive incorrect value

✓ **- 1 pts** *Incorrect Scanner instantiation (should be from the input file and from a File object)*

✓ **- 1 pts** *Incorrect PrintWriter instantiation (should be from the output file and from a File object)*

- 1 pts No loop iterating over the input file's line

✓ **- 1 pts** *Incorrect or missing check for odd lines*

- 1 pts Incorrect or missing line that writes to the output file one of the input file's lines

✓ **- 0.5 pts** *Doesn't close Scanner or PrintWriter*

✓ **- 0.5 pts** *Minor syntax error*

- 12 pts Incorrect / Missing

- 1 *Point adjustment*

Catches RuntimeException which doesn't let IOException propagate

1 -0.5 missing }

2 lineNumber is always 1

QUESTION 15

15 EC **1 / 1**

✓ **- 0 pts** *Correct*

- 1 pts No signature

CS 1331 – Exam 3

Fall 2022

Set 1

Name: Vidit Pokharna GTID: 903772087 Section: 01

GT username (i.e. gtg, gth, msmith3, not an alias): vpokharna3

By taking this exam, you signify that it is your work and that you have neither given nor received inappropriate help during the taking of this exam in compliance with the Academic Honor Code of Georgia Tech.

Signature: Vidit

(you must sign this for your exam to be graded.)

Your grade will be the percentage of points earned out the total points.

Check to make sure your exam has 8 pages. Each page has its number and the total number of pages on the bottom left corner. Your exam will be graded as submitted.

True/False [10pts, 2.5pts each] (approx. 2 minutes)

In each of the blanks below, write "T" if the statement beside the blank is true, "F" otherwise.

1) T More than one catch block can be executed if a try block throws an exception that matches the parameter type of more than one catch block.

2) F The Java compiler will generate errors for an interface that does not define the body of each declared method.

3) F Exceptions can be classified into three kinds: checked exceptions, unchecked exceptions, and terminating exceptions.

4) T Under normal circumstances, the finally block will always execute, even when there's a return statement or a thrown exception in all try blocks and catch blocks above it.

Multiple Choice [10pts] (approx. 3 minutes)

5) [3pts] If you were going to catch an exception of the Exception class, where should this catch block be placed within your code? (circle only one)

- Within the finally block System
- (•) Last among the list of catch blocks for a given try block
- There is no Exception type in Java
- First among the list of catch blocks for a given try block

6) [3ps] Java allows methods to call other methods. To keep track of method calls, Java utilizes a(n) … (circle only one)

- array of MethodCall instances
- (•) Call stack
- Call heap
- None of the above options

7) [4pts] A difference between the Exception and RuntimeException classes is that … (circle only one)

- (•) instances of Exception are thrown at compile-time, while instances of RuntimeException are thrown at runtime
- RuntimeException is used when a critical error occurs, while Exception is used for unexpected but less-critical encounters
- Java doesn't enforce the handling of exceptions extending from RuntimeException
- RuntimeException is generally used for programmer-created exceptions, while Exception are generally reserved to be thrown by the classes provided by Java

Short Response and Tracing [50pts] (approx. 25 minutes)

8) [4pts] Perform Binary Search in this array searching for the value 5. Indicate below the elements that are accessed when they are accessed (starting with 1 for the 1st element accessed, 2 for the second, etc.)

Array:	1	3	5	6	8	9	10
		2	3	1			

9) [5pts] Perform the first pass of Insertion Sort on the following array

Array:	3	2	8	1	6
Array after pass:	2	3	8	1	6

10) [4pts] Indicate the Big-O Notation and associated growth rate for Merge Sort and Selection Sort

Merge Sort Big-O Notation: $O(n \log(n))$

Merge Sort Growth Rate: Linearithmic

Selection Sort Big-O Notation: $O(n^2)$

Selection Sort Growth Rate: Quadratic

11) [4pts] Between Merge Sort and Selection Sort, indicate which is most suitable for long arrays (in the millions or billions of elements) and why.

merge sort because it has a growth rate that grows at a slower rate with increasing values of n (number of values in array) as compared to Selection sort and therefore would be more time efficient.

12) [3pts, 0.5pts each] Complete the table for the declared type and object type of the local variables (after the code executes, which always does successfully):

```
A v1;
B v2 = new B();
v1 = v2;
C v3 = new A();
```

Local Variable	Declared Type	Object Type
v1	A	B
v2	B	B
v3	C	A

13) [18pts, 3ts each] Review the code below, which compiles and executes without errors in its current state.

```
public class A {
    public String toString() { return "I am A";}
    public String stringA() { return "1";}
}
public class B extends A {
    public String toString() { return "I am B";}
    public String stringB() { return "2";}
}
public class Exam3 {
    public static void main(String[] args) {
        A a = new A();
        B b = new B();
        //*****You'll replace this line with individual statements below
    }
}
```

Now, with each statement below, replace the commented line (with asterisks) in Exam3's main method and indicate the result of the change. More specifically, you must indicate one of the following:

- the output of the code, if the code runs properly
- the exception thrown by the code, if the code compiles but doesn't run properly
- why it doesn't compile, if the code doesn't compile

A) `System.out.print(b.stringA());`

1

B) `System.out.print(((Object)a).stringA());`

Illegal Argument Exception

C) `System.out.print(((A)b).toString());`

I am A

D) `System.out.print(((B)a).toString());`

Class Cast Exception

E)
```
A v = (A) new B();
System.out.print(((B)v).stringB());
```

2

F)
```
B v = new A();
System.out.print(v.stringB());
```

This will not compile because the first line does not pass the "is a" check

14) [12pts, 4pts each] Complete the table for the declared type and object type of the local variables (after the code executes):

```
import java.io.*;
public class MyException extends Exception {
    private int value;
    public MyException(int value) {
        this.value = value;
    }
    public int getValue() {
        return(value);
    }
}

public class ExamThree {
    public static void foo(int x) throws MyException, IOException {
        if(x < 0) {
            System.out.print("A");
            throw(new MyException(x));
        }
        if(x > 100) {
            System.out.print("B");
            throw(new IOException("C"));
        }
        System.out.print("D");
    }
    public static void main(String[] args) {
        int x = Integer.parseInt(args[0]);
        try {
            foo(x);
            System.out.print("E");
        } catch(IOException e) {
            System.out.print("F");
        } catch(MyException e) {
            System.out.print("G");
            System.out.print(e.getValue());
        }
    }
}
```

a) Below, provide the expected output of: `java ExamThree -200`

AEG-200

b) Below, provide the expected output of: `java ExamThree 200`

BEF

c) Below, provide the expected output of: `java ExamThree 2`

DE

Coding [30pts] (approx. 20mins)

15) [8pts] Below is the int-based selectionSort method from the modules:

```
public static void selectionSort (int[] list) {
    int mindex;
    int nextSmallest;
    for (int unSortedStart=0; unSortedStart<list.length-1; unSortedStart++) {
        mindex = unSortedStart;
        for (int currIndex=unSortedStart+1; currIndex<list.length; currIndex++) {
            if (list[currIndex] < list[mindex]) {
                mindex = currIndex;
            }
        }

        nextSmallest = list[mindex];
        list[mindex] = list[unSortedStart];
        list[unSortedStart] = nextSmallest;
    }
}
```

Fill in the 3 blanks below so that it is a Comparable-based Selection Sort method:

```
public static void selectionSort (____A____[] list) {
    int mindex;
    ____B____ nextSmallest;
    for (int unSortedStart=0; unSortedStart<list.length-1; unSortedStart++) {
        mindex = unSortedStart;
        for (int currIndex=unSortedStart+1; currIndex<list.length; currIndex++) {
            if (____C____) {
                mindex = currIndex;
            }
        }

        nextSmallest = list[mindex];
        list[mindex] = list[unSortedStart];
        list[unSortedStart] = nextSmallest;
    }
}
```

A: Comparable

B: Comparable

C: list[currIndex].compareTo(list[minIndex]) < 0

16) [3pts] Write an interface called ExamInterface. It will have a single abstract method named "process" and it takes a String as a parameter. The method has no return value.

```
public interface ExamInterface {
    void process(String s);
}
```

17) [3pts] Write the header for a concrete class called Turkey that is: (a) a child of Bird and (b) a Comparable class. Make sure that the header uses a parameterized version of Comparable so that only Turkeys can be compared with one another.

```
public class Turkey extends Bird implements Comparable<Turkey>{}
```

18) [4pts] Write a compareTo method for Turkey that uses a wattleSize instance variable to determine order. A Turkey is greater than another if its (int) wattleSize is greater. Remember that the Turkey class implements a parameterized Comparable interface. You don't have to add code for the instance variable.

```
public int compareTo(Turkey other){
    return (wattleSize - other.wattleSize);
}
```

19) [12pts] Write a complete Java program called OddToAnother which receives two filenames from console arguments (none with whitespace) and copies the lines in odd line numbers from the first file to the second file. Line numbers start at 1 (so copy the 1st, 3rd, etc. lines to the second file). Let all runtime exceptions related to I/O propagate.

```
import java.util.*;
import java.io.*;
public class OddToAnother {
    public static void main(File[] args) throws IOException {
        Scanner scan = new Scanner(System.in);
        File f1 = new File(args[0]);
        File f2 = new File(args[1]);
        try {
            int lineNumber = 1;
            while(f1.hasNextLine()){
                String s = f1.nextLine();
                if(lineNumber % 2 == 1) {
                    f2.nextLine() = s;
                }
            }
        } catch(RunTimeException rte) {
            System.out.println(rte.getMessage());
        }
    }
```

20) [1pt EXTRA CREDIT] Make sure your exam has all the pages. Sign below to confirm your exam has all 8 pages.

Vidit Agrawal

Check to make sure your exam has 8 pages. Each page has its number and the total number of pages on the bottom left corner. Your exam will be graded as submitted.