Name (print clearly): _____

9-Digit GTID: _____ Section (e.g, A1): _____

Problem	Type	Points Possible
1) Foundation	Multiple Choice and T/F	25
2) Recursion	Coding	16
3) Exceptions	Tracing	12
4) JavaFX	Tracing	10
5) Interfaces and Abstract Classes	Multiple Choice	10
6) Generics	Coding	15
7) File IO	Tracing	12
Bonus #1		3
Bonus #2		2
TOTAL		100 + 5

Please remember: Any academic misconduct (including, but not limited to, the list below) could result in a 0 (zero) on the exam and/or an F grade in the course:

- Communication with anyone other than a proctor for ANY reason in ANY language.
- Sharing of ANYTHING (e.g. pencils, paper, erasers).
- Writing on paper that is not given to you by a proctor.
- Failure to follow directions given by the proctor.
- Use of cell phones, beepers, handheld computers, calculators, during the exam.
- Using books or other reference material.
- Disruption of the exam setting.
- When you turn in your exam, you will have to show your ID to the TAs before we will accept your exam. It is your responsibility to have your ID prior to beginning the exam.
- You are not allowed to leave the exam room and return. If you leave the room for any reason, then you must turn in your exam as complete.
- Notes, books, calculators, phones, laptops, smart watches, headphones, etc. are not allowed.
- Extra paper is not allowed. If you have exhausted all space on this test, talk with your instructor. There are extra blank pages in the exam for extra space.
- Style standards such as (but not limited to) use of good variable names and proper indentation is always required. If it is unclear which letters are capital and lowercase, underline the capital letters, ex: SomeName
- Comments are not required unless a question explicitly asks for them.

HINT: Once the exam begins, skim through it first.

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: _____

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

This page is intentionally left blank. Do not write ANY answers here, go to the next page.

(25 pts) 1. **Multiple Choice and T/F**

Answer the following questions by filling in the circle **completely** next to the correct answer.

(2 pts) (a) **T/F:** A variable's static type can be an interface.

① True

② False

(2 pts) (b) **T/F:** A JavaFx FlowPane arranges the nodes in a grid specified by the program writer that does not change as the window is resized.

① True

② False

(2 pts) (c) **T/F:** Every subclass of `RuntimeException` is a checked exception.

① True

② False

(2 pts) (d) **T/F:** An interface can extend an interface.

① True

② False

(2 pts) (e) **T/F:** In recursion, the base case can be a recursive call.

① True

② False

(3 pts) (f) Analyzing algorithm efficiency using Big-O is _____ .

① to measure their actual execution time

② to estimate their growth function

③ to estimate their execution time

(3 pts) (g) Which of these runtimes is the **worst**?

① $O(n^2)$

② $O(3*n)$

③ $O(1)$

④ $O(n^n)$

(3 pts) (h) To declare an interface named A with **two generic types**, use _____ .

① `public interface A(E, F) { ... }`

② `public interface A<E> { ... }`

③ `public interface A(E) { ... }`

④ `public interface A<E, F> { ... }`

(3 pts) (i) Suppose `A` is an interface, and `B` is a concrete class with a no-arg constructor that implements `A`. Which of the following are correct? (**mark all the apply**)

① `B b = new A();`

② `B b = new B();`

③ `A a = new A();`

④ `A a = new B();`

(3 pts) (j) For a binary search to work correctly, the list being searched must meet the following condition(s) (**mark all that apply**)

① only contain numbers

② contain objects

③ be sorted

(16 pts) 2. **Recursion**

Implement the following problem recursively. Given a non-negative `int n`, return the sum of its digits recursively (no loops). For example:

- sumDigits(126) gives 9
- sumDigits(49) gives 13
- sumDigits(12) gives 3

Note:

- n mod (%) by 10 yields the rightmost digit of n (e.g. 126 % 10 is 6)
- n divide (/) by 10 removes the rightmost digit of n (e.g. 126 / 10 is 12).

```
public class RecursiveQuestion {
    /* This is your recursive method */
    public static int sumDigits(int n) {









    }
}
```

(12 pts) 3. **Exceptions**

Consider the code below. Assume that all needed imports are included.

```
// assume the main method below is inside of a public class
public static void main(String[] args) {
    try {
        System.out.println("Go long!");
        /* ~~~ Some code will be inserted here ~~~ */
        System.out.println("What was that?!");         // Line 1
    } catch (Baseball b) {
        System.out.println(b.getMessage());           // Line 2
        System.out.println("They're out!");           // Line 3
    } catch (Ball b) {
        System.out.println(b.getMessage());           // Line 4
        System.out.println("Nice catch!");            // Line 5
    } catch (Throwable t) {
        System.out.println(t.getMessage());           // Line 6
        System.out.println("You're pretty good!");    // Line 7
    } finally {
        System.out.println("The game's over folks!"); // Line 8
    }
    System.out.println("Thanks for playing!");        // Line 9
}

class Ball extends Throwable {
    public Ball(String message) {
        super(message);
    }
}

class Baseball extends Ball {
    public Baseball(String message) {
        super(message);
    }
}
```

For each of the following independent conditions, write the line numbers that will print (use the line numbers commented on the right side). For example: `1, 2, 4, 9`

(4 pts) (a) `throw new Throwable("Catch!");` replaces the comment in the try block.

(4 pts) (b) `throw new Ball("Ultimate Frisbee!");` replaces the comment in the try block.

(4 pts) (c) `throw new Baseball("They're trying to steal home!");` replaces the comment in the try block.

(10 pts) 4. **JavaFX**

Consider the code below. Suppose the program has started and is displayed on the screen. Draw a picture of what the GUI window will look like after the user clicks the "Click Me" button **3 times**.

```
1  // assume all necessary imports are included and correct
2  public class JavaFxQuestion extends Application {
3      private VBox vPane = new VBox(25.0);
4
5      @Override
6      public void start(Stage primaryStage) throws Exception {
7          Rectangle rectangle = new Rectangle(100, 100);
8          vPane.getChildren().add(rectangle);
9          vPane.setAlignment(Pos.CENTER);
10
11         Button myButton = new Button("Click Me");
12         myButton.setOnAction(new ButtonHandler());
13         vPane.getChildren().add(myButton);
14
15         Scene myScene = new Scene(vPane, 300, 300);
16         primaryStage.setScene(myScene);
17         primaryStage.show();
18     }
19
20     class ButtonHandler implements EventHandler<ActionEvent> {
21         @Override
22         public void handle(ActionEvent event) {
23             Node firstNode = vPane.getChildren().get(0);
24             // setRotate(double rotateDegrees)
25             firstNode.setRotate(firstNode.getRotate() + 45.0);
26         }
27     }
28 }
```

(10 pts) 5. **Interface and Abstract Classes**

```
1  public interface Foo {
2      void biz();
3
4      default void bill() {
5          System.out.println("bill");
6      }
7  }
8
9  public abstract class Jam {
10     public void jar() {
11         System.out.println("jar");
12     }
13     public abstract void jill();
14     private void jazz() {
15         System.out.println("club");
16     }
17 }
```

(5 pts) (a) Given a class that extends `Jam` and implements `Foo`, choose **ALL** methods that **MUST** be overridden.(**mark ALL that apply**)

① biz
② bill
③ jar
④ jill
⑤ jazz

(5 pts) (b) Given a class that extends `Jam` and implements `Foo`, choose **ALL** methods that **CAN** be overridden. (**mark ALL that apply**)

① biz
② bill
③ jar
④ jill
⑤ jazz

(15 pts) 6. **Generics**

Implement a generic class called `Box` below. Instances of `Box` should be able to contain instance data of whatever is passed in for the generic type parameter. The generic type must be an `Animal` or a sub-type of `Animal`. To complete the class, you must do the following:

- Finish the class declaration.
- Make an instance variable of the generic type that is not accessible to outside classes and is called `data`. The `data` should also be `final` (immutable) .
- Make a publicly available constructor that initializes the instance variable to the `data` parameter passed in.
- Make a getter method for the instance variable that utilizes the standard naming convention (`getData`).

```
public class_____________________________________________ {




}
```

(12 pts) 7. **File IO**

Consider the code below.

```
1  public static void main(String[] args) {
2      try {
3          Scanner scan = new Scanner(new File("text.txt"));
4          PrintWriter writer = new PrintWriter("someFile.txt");
5          int num = 0;
6          while (scan.hasNextLine()) {
7              num++;
8              String str = scan.nextLine();
9              if (str.contains("hello")) {
10                 writer.println(num + ": " + str.toUpperCase());
11                 writer.flush();
12             }
13         }
14     } /* ~~~ Missing Code ~~~ */ {
15             ex.printStackTrace();
16     }
17 }
```

(8 pts) (a) Briefly explain (in 2 sentences or less) what the `main` method does; i.e. what functionality is achieved. Do **NOT** give a line-by-line description of the method.

(4 pts) (b) Write what should be on line 14 in order to make this code to compile. Be as specific as possible!

(3 pts) 8. **Bonus #1**

Assume you are creating a new unchecked exception, called `NewException`.
You want to create a method, named `m1`, that takes in either an instance of `NewException`, any subclasses of `NewException`, or any other exceptions that the Java compiler will not worry about at compile-time. What is the correct method header definition?

① `public void <T extends Throwable> m1 (T someException) ...`

② `public void <T extends RuntimeException> m1 (T someException) ...`

③ `public void <T extends NewException> m1 (T someException) ...`

④ `public void <T extends Exception> m1 (T someException) ...`

(2 pts) 9. **Bonus #2**

What was something fun you did on break?

Congratulations on completing your exam! Make sure you have completed the first page (including signing it) and turn into a TA.