Exam 3 Prep

Tony Peng

Good luck !

Multiple choice

1. How many of the following are checked?

   I: IllegalArgumentException

   II: FileNotFoundException

   III:NullPointerException

   IV: Exception

   V: OutOfMemoryError

   VI: Throwable

A. 1
B. 2
C. 3
D. 4
E. 5

2. Which of the following is a **FALSE** statement about the interface Comparable?

   A. Comparable has exactly one abstract method.
   B. Comparable will return 1 if the object is greater than the other one comparing to, -1 if it's less.
   C. Comparable's method compareTo compares two objects based on their natural ordering, assuming both objects are instances of Comparable
   D. Collections.sort() utilizes the natural ordering provided by Comparable's method compareTo.
   E. None of the above are false statements.

3. What will the following code print?

```
ArrayList<String> arr = new ArrayList<>(5);
for(int i = 0; i < 10; i++){
      arr.add("placeholder");
}
```

   A. Compile error
   B. Exception at thread main: IndexOutOfBoundsException….

C. Nothing
D. None of the above
E. All of the above

4. How many of the followings are a valid statement?
   I. E temp;
   II. E temp = new E();
   III. E[] arr = new E[10];
   IV. (E) temp;
   V. temp instanceof E;

   A. 1
   B. 2
   C. 3
   D. 4
   E. 5

5. When is a check exception thrown?
   A. It depends
   B. Runtime
   C. Compile time
   D. Since you must catch a checked exception and handle it, so never.
   E. None of the above

6. What of the following can you **throw** in the code?
   A. VirtualMachineError
   B. FileNotFoundException
   C. IllegalArgumentException
   D. B and C
   E. All of the above

7. What of the following can you **catch** in the code?
   A. VirtualMachineError
   B. FileNotFoundException
   C. IllegalArgumentException
   D. B and C
   E. All of the above

8. Which of the following about throw vs throws is/are true?
   I. If method2 has a possibility of throwing a checked exception and it's not caught, the method2 header has to declare a throws statement.

II. If method2 has a possibility of throwing a unchecked exception and it's not caught, the method2 header has to declare a throws statement.
III. When an exception is thrown, your code is going to crash.
IV. Throw is used when you want to manually generate an exception that may or may not crash the code.
V. When you throw a unchecked exception, you generally want the client code to crash.

A. I, II, V
B. I, V
C. I, IV, V
D. II, III, IV, V
E. All of the above

9. If InsufficientSleepException directly extends Exception, what type of Exception will InsufficientSleepExecption be?

A. Error
B. Checked Exception
C. Un-Checked Exception
D. B and C
E. Cannot determine

10. Which of the following is not a Collection?

A. Set
B. List
C. Map
D. They are all not Collection
E. They are all Collection

Short Answer

1. Which package is Comparable interface in?

2. Which package is Collections in?

3. Instantiate a set that takes in String values and store it inside a variable.

4. What is the purpose of generics?

5. If you have an ArrayList that contains a series of Integers. Write a line of code to sort the numbers in order.

6. What is functional interface, and give two examples of them.

7. If you have MyArrayList<Person> list = new MyArrayList<>(). What is the precondition if you want to use a for-each loop on the list.

8. If you have MyArrayList<Person> list = new MyArrayList<>(), What is the precondition if you want Collections.sort(list) to work.

9. If a class implements Comparable<Student>. What method do you have to override, write down the whole method header?

10. If a class implements Iterable<Student>. What method do you have to override, write down the whole method header?

11. You have the following code:
    ```
    ArrayList<Integer> list = new ArrayList<>();
    list.add(new Integer(2))
    list.add(1);
    list.add(new Integer(3));
    list.add(4);
    list.add(new Integer(5));
    ```

    a. What does the list look like now?

    b. now we execute list.remove(2), what does the list look like now?

    c. How about now ? list.remove(new Integer(1));

12. Without changing directories, run the main method within the HelloWorld class located in "home/user/java" (assume you aren't in the same directory)

13. Standard path for java source files (src/main/java) or Standard path for compiled class files (src/target/classes)

14. What line would you include at the top of a java file to add it to the edu.gatech.cs1331 package?

15. Write a line to execute the HelloWorld.jar file

Fill in the blank to indicate which data structure you should use in the following scenarios.

A. Set. B. List. C. Map

1. _____ You want to keep track of who is coming to a party that you are throwing tonight.

2. _____You want to keep track of every students in the class and look up a student's name using their GTID.

3. _____You are writing a game and you want to keep track of the inventory items a player has gathered based on the time acquired.

4. _____You want to keep track of the food that you are allergic to.

5. _____You want to write a dictionary that keep tracks of vocabulary and the definition.

6. _____Georgia tech wants to keep track of every items that is over 1000 dollars throughout the campus.

Tracing

1.

```
try{
    int a = 5;
    int b = 0;
```

```
        System.out.println("Let's begin");
        System.out.println(a/b);
        System.out.println("this doesn't work");
} catch(IllegalArgumentException e){
        System.out.println("illegal argument");
} catch(Exception e){
        System.out.println("some exceptions");
        throw new NullPointerException("error");
} finally {
        System.out.println("Hello World");
}
```

2.

```
public static void main(String[] args){
        System.out.println(method1(5));
        System.out.println();
        System.out.println(method1(4));
}
public static int method(int n){
        try{
            if(n%2 == 1)
                    throw new NullPointerException("error");
            else
                    throw new IOException("IOError");
            } catch (IOException){
                    System.out.println("caught IO");
                    return 3;
```

```
        } finally {
                return 5;
        }
}
```

3. You have the following code

```
public class Car {
        public  void calculateVroom(int b) throws Exception{
                System.out.println("Speed : " + b);
                if(b < 0) help();
                System.out.println("Goin' vroom");
        }

        public static void help() throws Exception {
                throw new Exception("No vroom");
        }
}
```

a. What would the following code print?

```
public static void mian(String[] args) throws Exception {
        Car c = new Car();
        try{
                System.out.println("Rev!!");
                c.calculateVroom(-4);
                System.out.println("faster!");
        } catch (Exception e) {
                e.getMessage();
        }
}
```

b. What would the following code print?

```
public static void main(String[] args) throws Exception {
```

```
        Car c = new Car();
        try{
                System.out.println("Go!");
                c.calculateVroom(1);
                System.ou.println("Vroom!");
        } catch (Exception e) {
                e.getMessage();
        } finally {
                System.out.println("Phew!");
        }
}
```

Coding

1. Complete the following class so that Person correctly implements the Comparable interface. A Person should be sorted in an ascending order based on age, if the age is the same, sort them by the name.

   ```
   public class Person implements _____{

           public String name;
           public int age;
           public Person(String name, int age){....}

           // Your code here



   }
   ```

   Now write a comparator class called PersonComp that compares the person by the age only.

```
public class PersonComp implements _____{

        //Your code here




}
```

After wrting the comparator class we can do something like this

```
ArrayList<Person> list = …..;
Collections.sort(list, new PersonComp());
```

Now convert Collections.sort(list, new PersonComp()) into an anonymous class.

Now convert Collections.sort(list, new PersonComp()) into lambda expression.

2. Write a custom Exception called OutOfMoneyException, it should be a checked exception, and should take in an error message upon throwing.