Eldor Bekpulatov (eb654)

1. Answer the following short questions

(a) Which Java class is the parent class of all Exceptions?

Throwable is the parent class of all Exceptions.

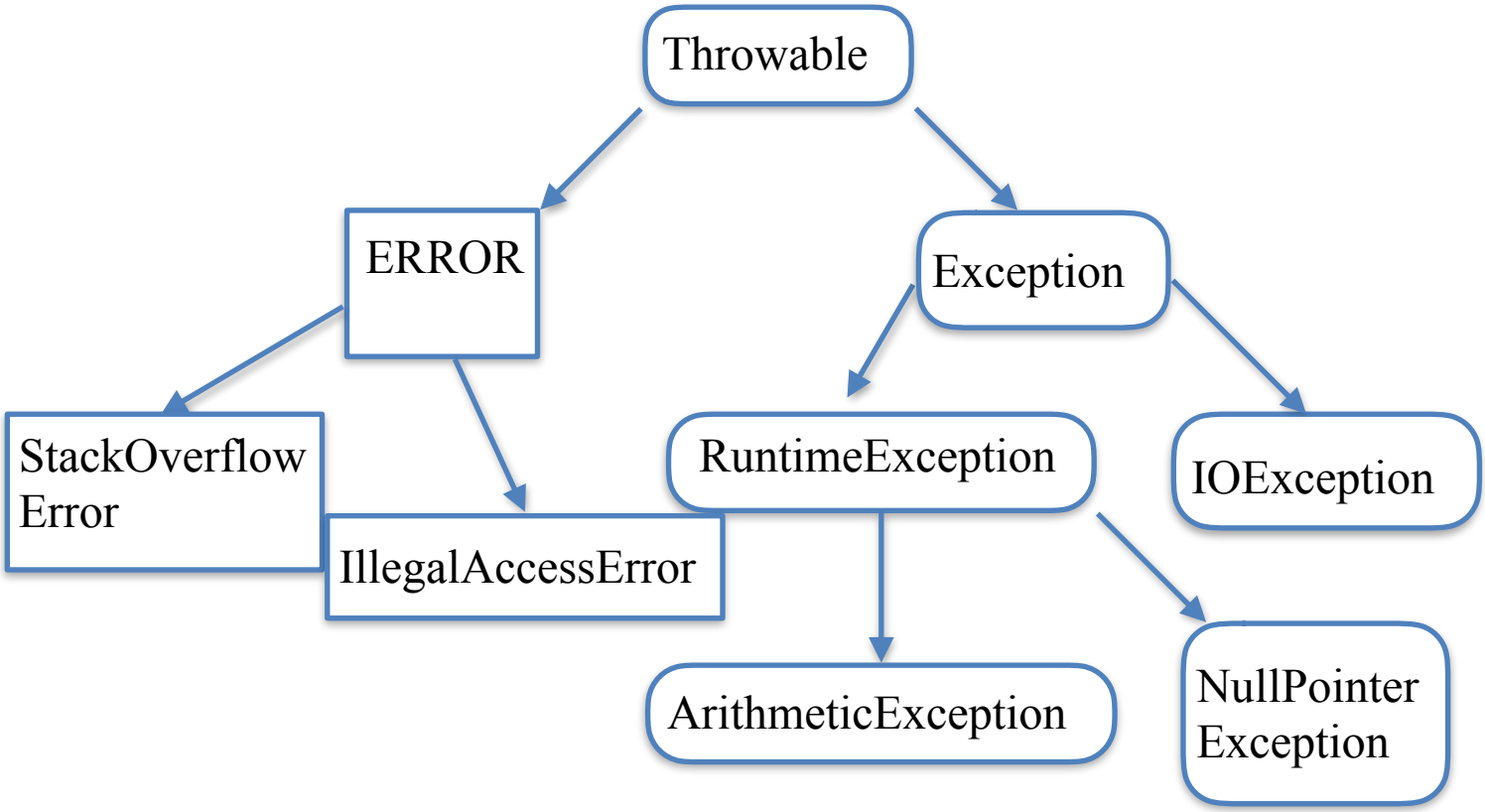
(b) What is the difference between Exceptions and Errors?

Exceptions are throwable objects that can be handled and may cause the program to terminate. Whereas, Errors are throwable objects that aren't designed for the program to handle, so the program terminates.

(c) What is the purpose of Exceptions? Give an example of what they're used for.

The purpose of Exceptions is to allow the program recover from variety of error cases. Exceptions are labeled with specific types, so that when certain expected errors are thrown, the program can have fail-safe way of recovering itself. One example that I can think of is the division by zero exception. So, when you have a function that returns the quotient of two numbers, and the second number is zero, Division by Zero exception can be caught by the program and ask the user to input valid numbers.

2. Draw the basic exception-class hierarchy, with Throwable at the top. Include at least two subclasses of Error, Exception, and RuntimeException.



3. In the following statements, suppose S0 throws an Exception, so S1 is executed. State what happens in two cases:

(1) S1 throws an exception
It would get caught by the program.
S3 would not be executed.
(2) S1 does not throw an exception.
It would execute S1 then S3.

```
try { S0 }
catch (Exception e) { S1 }
S3
```

4. Write a throw statement that throws an ArithmeticException with message "*arg should not be negative*".

Remember: use a new-expression to create a throwable object.

```
throw new ArithmeticException("arg should not be negative”);
```

5. Consider the following class:

```
public class A {
     public static double p(int x) {
          int y= x;
          try {
               System.out.println("six");
               y= 5/x;
               System.out.println("five");
               return 5/(x + 2);
          } catch (RuntimeException e) {
               System.out.println("four");
               y= 5/(x+1);
               System.out.println("three");
          }
          System.out.println("two");
          y= 4/x;
          System.out.println("one");
          return 1/x;
     }
}
```

(a) Below, write what is printed by execution of the call p(0).

six
four
three
two
java.lang.ArithmeticException: / by zero

(b) Below, write what is printed by execution of the call p(-2) .

six
five
four
three
two
one

(c) Below, write what is printed by execution of the call p(-1) .

six
five

6. Here is a method to return the minimum value of array segment c[m..n]. It is not correct because it does not throw the required exception if c[m..n] is empty. Place a suitable throw statement at the beginning of the method body. You can assume that c is not null.

```
/** Return the minimum value in c[m..n]. Throw a RuntimeException with
       * message "min of 0 values doesn't exist" if c[m..n] is empty. */
      public int min1(int[] c, int m, int n) {
            if (c.length==0){
            throw new RuntimeException("min of 0 values doesn't exist");
            }
            int min= c[m];
            for (int k= m+1; k <= n; k= k+1) {
                  if (c[k] < min) min= c[k];
            }
            return min; }
```

7. Here is a method to return the minimum value of array c It is not correct because it does not throw the required exception if c is empty. C is empty if its length, given by c.length, is 0. Yes, it can be 0; you can create an array with 0 values: int[] c= new c[0].

```
/** Return the minimum value in c. Throw a RuntimeException with
 * message "min of 0 values doesn't exist" if c is empty. */
public int min2(int[] c) {
      int min= c[0];
      for (int k= 0; k < c.length; k= k+1) {
            if (c[k] < min) min= c[k];
      } return min; }
```

Instead of inserting a throw statement in the method body, below, rewrite the method body to use a try statement, with the whole method body in the try block. Let the catch clause catch an ArrayIndexOutOfBoundsException. In the catch block, throw the required exception. You don't have to copy the method specification or header; just write the method body:

```
try {
     int min= c[0];
     for (int k= 0; k < c.length; k= k+1) {
          if (c[k] < min) min= c[k];
     }
}
catch (ArrayIndexOutOfBoundsException e){
    throw new RuntimeException("min of 0 values doesn't exist”);
}
return min;
```

8. Consider the following class. Several possible sequences of numbers can be printed by a call first(i); depending on the value of i. List each sequence of numbers that can be printed by such a call, along with the value of i that causes that sequence to be printed.

```
public class A {
     public static void first(int k) {
          try {
               System.out.println("0");
               second(k);
               System.out.println("1");
          } catch (Exception e) {
               System.out.println("2");
          }
     }
     public static void second(int p) throws Exception {
          System.out.println("3");
          try {
               int b= 5/p;
               System.out.println("4");
               if (p == 6) throw new Exception();
               System.out.println("5");
          } catch (ArithmeticException e) {
               System.out.println("6");
          }
          System.out.println("7");
     }
}
```

List of Inputs and Outputs:

first(0);	first(1);	first(6);
0	0	0
3	3	3
6	4	4
7	5	2
1	7	
	1	

9. Consider class C given below. Function Integer.parseInt throws a NumberFormatException if its argument does not contain an integer. Below class C, rewrite the class so that if Integer.parseInt throws an exception, the number 1 is used. Note that Integer.parseInt is called in two places so you may need two try-statements. Don't be concerned with how one reads from the keyboard, pausing until something is typed.

```
public class A {
  /** Print the sum of two integers read from the keyboard */
  public static void main(String[] args){
     try {
          System.out.println("Enter a number: ");
          String s;
          //Read a line from the keyboard and store it in s;
          int a= Integer.parseInt(s);
     } catch (NumberFormatException a){
          a=1;
     }
     try {
          System.out.println("Enter another number: ");
          //Read a line from the keyboard and store it in s;
          int b= Integer.parseInt(s);
     } catch (NumberFormatException b){
          System.out.println("Product: " + a*b);
     }
  }
}
```