Exception name = type of error + “Exception”

NullPointerException

DivideByZeroException

DiskAccessException

Implement constructor method:

- Default constructor
- Single-argument constructor: error message
- Invokes suitable super constructors

```
public class NonPositiveException extends RuntimeException (unchecked)
{
      public NonPositiveException(String msg) {
            super(msg);
      }
}
```

Specification: throw exceptions

```
/**
* @effects <pre>
*     if n is non-positive,
*           throws NonPositiveException
*     else
*           returns the factorial of n
* </pre>
*/
public static int fact(int n) throws NonPositiveException
```

```java
/**
* @requires <tt> a </tt> is sorted
* @effects <pre>
*     if a is null,
*           throws NullPointerException
*     else if x is not in a
*           throws NotFoundException
*     else
*           returns i such that a[i] = x
* </pre>
*/
public static int search(int[] a, int x) throws NullPointerException,
                                                  NotFoundException
```

```java
/**
* @effects <pre>
*     if n is non-positive,
*           log error and  return -1
*     else
*           returns the factorial of n
* </pre>
*/
public static int computeFact(int n) {
      try {
            int f = fact(n);
            return f;
       } catch (NonPositiveException e) {
            System.err.println(“Error: invalid input” + e.getMessage());
            return -1;
      }
}
```

Reflect an exception

```
/**
* @effects <pre>
*     if n is non-positive,
*           throw NotPossibleException
*     else
*           returns the factorial of n
* </pre>
*/
public static void computeFact(int n) throws NotPossibleException {
      try {
            int f = fact(n);
            System.out.println(fact(n): f);
      } catch (NonPositiveException e) {
            throw new NotPossibleException("Could not compute fact(n)");
      }
}
```

Partial procedure

```
/**
* @requires <tt> a != null </tt>
* @effects <pre>
*     if a is sorted in ascending order,
*          return true
*     else
*          return false
* </pre>
*/
boolean sorted(int [] a)
```

Total procedure

```
/**
* @effects <pre>
*     if a is null,
*          throws NullPointerException
*     else if a is sorted in asc order
*          return true
*     else
*          return false
* </pre>
*/
public static boolean sorted(int[] a) throws NullPointerException
```

Overusing

```
/**
* @requires <tt> a != null </tt>
* @effects <pre>
*     if a is sorted in ascending order,
*          return true
*     else
*          return false
* </pre>
*/
boolean sorted(int [] a) {
     int prev;
     try {
          prev = a[0];
     } catch (IndexOutOfBoundsException e) {
          return true;
     }
     for (int i = 1; i < a.length; i++) {
          if (prev <= a[i])
               prev = a[i];
          else
               return false;
     }
     return true;
}
```

Commented [WU1]: