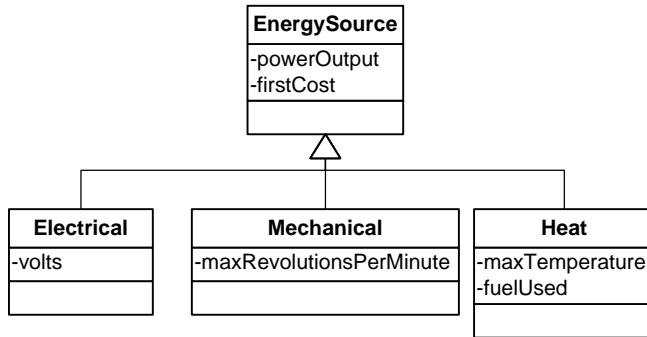


CSCE 145 Sections 3&4: Test 2 Key, Fall 2011

1. Write the code for a public static method *min* that takes three integers as parameters and returns the smallest value of the three.

```
public static int min (int a, int b, int c) {  
    if (a < b && a < c) return a;  
    else if (b < c) return b;  
    else return c;  
}
```

2. Suppose you want to create a computer description of various kinds of energy sources, including the four classes *Electrical*, *EnergySource*, *Mechanical*, and *Heat*, and the six instance variables *firstCost*, *fuelUsed*, *maxRevolutionsPerMinute*, *maxTemperature*, *powerOutput*, and *volts*. Draw a UML class diagram with class names, variable names, and inheritance arrows, showing which class should get each variable and which classes inherit from each other.



3. What are your three initials? MNH Write a public method called *removeInitials* that has an array of characters as input and that returns the array of characters left after your initials have been removed. Assume that the input and your initials are in upper case.

```
public char[] removeInitials(char[] c) {  
    int j = 0; // the index of the output array  
    char[] d = new char[c.length]; // the output array  
    for (int k = 0; k < c.length; k++) { // k is the index of the input  
        if (c[k] != 'M' && c[k] != 'N' && c[k] != 'H') {  
            d[j] = c[k];  
            j++;  
        }  
    }  
}
```

4. Add try - catch blocks to the following program to make it compile and execute correctly, even when the divisor is zero. Note that division by zero throws an *ArithmeticException*.

```
public class Division {  
    public static void main(String[] args) {  
        Scanner kb = new Scanner(System.in);  
        int n, d, q;  
        System.out.print("Enter numerator: ");  
        n = kb.nextInt();
```

```

        System.out.print("Enter denominator: ");
        d = kb.nextInt();
    try {q = n / d;
        System.out.print(q);
    }
    catch(ArithmeticException e) {
        System.out.println(e.getMessage());
    }
}
}

```

5. [Multiple catch blocks] Suppose the code in a try block might throw any of the following exceptions: (a) Exception, (b) IllegalArgumentException, (c) IOException, (d) NumberFormatException, (e) RuntimeException. Identify the best sequence for multiple catch blocks of these types.
- (d) **NumberFormatException**
 (b) **IllegalArgumentException**
 (c) **IOException** (e) **RuntimeException**
 (a) **Exception**

6. Write the code for the method *min* that takes an array of doubles as its parameter and returns the largest double in the array.

```

public double min(double[] d) {
    double dMin = d[0];
    for (int k = 1; k < d.length; k++)
        if (d[k] < dMin) dMin = d[k];
    return dMin;
}

```

7. What results are printed when the *main* method in TestExceptions is executed?

```

public class TestExceptions {
    private double[] value = new double[] {1.0, 0.9, 0.8, 0.6, 0.4, 0.2};
    private int num;
    public double eval(String s1, String s2) throws
    IndexOutOfBoundsException {
        try {
            num = Integer.parseInt(s1) / Integer.parseInt(s2);
        }
        catch (NumberFormatException nfe) {
            num++;
            System.out.println("in first catch");
        }
        catch (ArithmeticException ae) {
            num++;
            System.out.println("in second catch");
        }
        return value[num];
    }
}

public static void main(String[] args) {
    TestExceptions te = new TestExceptions();
    try {
        System.out.println(te.eval("5.0", "4"));
        System.out.println(te.eval("5", "0"));
        System.out.println(te.eval("22", "5"));
    }
}

```

```

        System.out.println(te.eval("33", "5"));
    }
    catch (Exception e) {
        System.out.println("in main's catch");
    }
    System.out.println("Bye");
}
}
in first catch
0.9
in second catch
0.8
0.4
in main's catch
Bye

```

8. Write a method called `fileCompare` that has two parameters: the String names of two text files. Your method should open a stream to each of the files, read one line-at-a-time from each file, and compare the two lines. If the two lines are not equal, it should print the file name and its line in the console window. Else print nothing. For example, if the text files contain the following:

File1	File2
This is a test of the file comparison program. It is written in Java. It works perfectly. The End.	This is a test of the file comparison software. It is written in C++. It does not work. The End.

Then the result printed by the method should be

File1: program. It is written in Java.

File2: software. It is written in C++.

File1: It works perfectly.

File2: It does not work.

```

public static void fileCompare(String file1, String file2) {
    Scanner inStream1 = null;
    Scanner inStream2 = null;
    String s1 = null;
    String s2 = null;
    try {
        inStream1 = new Scanner(new File(file1));
        inStream2 = new Scanner(new File(file2));
    } catch (FileNotFoundException e) {
        System.out.println("Error opening the file " +file1);
        System.exit (0);
    }
    while (inStream1.hasNextLine() && inStream2.hasNextLine()) {
        s1 = inStream1.nextLine();
        s2 = inStream2.nextLine();
        if (!s1.equals(s2)) {
            System.out.println("File1: " +s1);
            System.out.println("File2: " +s2);
        }
    }
}

```