Final Report: Java Programming Language.

A simple project which use some methods such as: sorting, bubble search, filling, printing arrays and others.

Agata Gabara

Information Technology

Under supervision of Padma Daryanani

Middlesex University, London

ag1485@live.mdx.ac.uk

CONTENT

Introduction:

"Java is a general-purpose, concurrent, class-based, object-oriented computer programming language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another. Java applications are typically compiled to byte code (class file) that can run on any Java virtual machine (JVM) regardless of computer architecture. Java is, as of 2012, one of the most popular programming languages in use, particularly for client-server web applications, with a reported 10 million users"[1]

"Java supports object-oriented programming techniques that are based on a hierarchy of classes and well-defined and cooperating objects"[2]

"Inheritance and Polymorphism: [3]One object-oriented concept that helps objects work together is inheritance. Inheritance defines relationships among classes in an object-oriented language. The relationship is one of parent to child where the child or extending class inherits all the attributes (methods and data) of the parent class. In Java, all classes descend from java. lang".

Try and catch

When a Java program performs an illegal operation (division by zero, access an array at a position which does not exist, etc) an event known as exception happens. When an exception occurs, we say an exception is thrown.

Usually, when an exception occurs, the program will terminate immediately. However, Java provides ways to detect that an exception has occurred. This process is called exception handling.

Catching Exceptions
• try block: Method calls which may cause an exception can/must be put in a try block.
• catch block: the catch block indicates what should happen in case an exception occurred.

e

When an exception occurs, Java creates an exception object which (was called e in the last example and) contains information about the error. • Every exception object contains a string message, which can be used rather then printing your own message.

[1] "Programming Language Popularity". 2009. Retrieved 2009-01-16.
[2] Monica Pawlan , Essentials of the Java Programming Language A Hands-On Guide
[3] Mads Rosendahl, Introduction to graphics programming in Java, February 13, 2009

try { quot = a / b; }

catch (ArithmeticException e)

{ System.out.println(e.getMessage()) }

Do's and Don'ts

• Throwing exceptions is better than just passing a boolean flag whether an operation was successful. (The calling method simply might forget to check the flag.)

• Do throw specific Exceptions (and not just a RunTimeException if you mean something more specific).

• Throw early, Catch late. (It is better to declare that a method throws a checked exception than to handle the exception poorly.)

Example from lab:

```
package javaapplicationerrors;
import javax.swing.JOptionPane;
public class JAVAApplicationErrors {
public static void main(String[] args) {

   String str = JOptionPane.showInputDialog("please enter a number");
   try {
   int a = Integer.parseInt(str);
   } // end of try

    catch (NumberFormatException e) {
    System.out.print("NumberFormatException: ");
    System.out.println(e.getMessage());

 } // end of catch
} // end of main method
} // end of main class
```

```
put - JAVAApplicationErrors (run)
run:
BUILD SUCCESSFUL (total time: 5 seconds)
```

And another:

```
package example_ex;
import java.util.Scanner;
public class Example_ex {
public static void main(String[] args) {

    Scanner in = new Scanner(System.in);
    boolean b = true;
    while (b) {

    try {
    System.out.println("Please enter number");
    String str = in.next();
    int a = Integer.parseInt(str);
    b = false;

 } // try

catch (NumberFormatException e) {
System.out.println("Not an integer, try again");


 }// catch
 }// while
 } // main
}// class
```

Bubble sort

"Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order.

First Pass:
(5 1 4 2 8) –> (1 5 4 2 8), Here, algorithm compares the first two elements, and swaps since 5 > 1.
(1 5 4 2 8) –> (1 4 5 2 8), Swap since 5 > 4
(1 4 5 2 8) –> (1 4 2 5 8), Swap since 5 > 2
(1 4 2 5 8) –> (1 4 2 5 8), Now, since these elements are already in order (8 > 5), algorithm does not swap them.

Second Pass:
(1 4 2 5 8) –> (1 4 2 5 8)
(1 4 2 5 8) –> (1 2 4 5 8), Swap since 4 > 2

(1 2 4 5 8) –> (1 2 4 5 8)
(1 2 4 5 8) –> (1 2 4 5 8)"[4]

Example from lecture:

```
package testbubblejava;

import javax.swing.*;

import java.util.*;

public class TestBubbleJAVA {

public static void main(String[] args) {

                final int SIZE = 4;

                int numbers[] = new int[SIZE];

                fill_array (numbers, SIZE);

                print_array (numbers, SIZE); // print unsorted list

                bubble_sort (numbers, SIZE);

                print_array (numbers, SIZE);// prints after sorting

                System.exit(0);

   }

   // Sort an array of integers using (an inefficient) variant of

        // bubblesort

        public static void bubble_sort (int array[], int arraySize)

        {       int total=0; int x=1;

                for (int pass = 0; pass < arraySize - 1; pass++) //  happens 3 times only - controls inner loop

        {   for(int counter = 0; counter < arraySize -  x; counter++) // happens 4 times
```

[4] https://www.geeksforgeeks.org/bubble-sort/

```
            {if (array[counter] > array[counter+1])  // compare 2 elements in the array
          swap (array, counter, counter+1); // pass parameters
            //        total++; // number of loops
          print_array (array, arraySize); // print intermediate states as is sorted
            }
            x++;
      }
      }
      // Exchange a given pair of values given by their positions in an array
      public static void swap (int a [], int p1, int p2)
      {
      int temp;
     temp = a[p1];
      a[p1] = a[p2];
      a[p2] = temp;
      }
      // Fill an array from the keyboard
      public static void fill_array (int array[], int arraySize)
      {for(int counter = 0; counter < arraySize; counter++)
      {array[counter] = Integer.parseInt(JOptionPane.showInputDialog ("Number for
position:"+ counter));
      }
      }
      // Print the contents of an array to the screen
      public static void print_array (int array [], int arraySize)
```

```
        {   int counter=0;


 for(counter= 0; counter < arraySize; counter++)
                {
   System.out.print(array[counter] + " \t");
                }
   System.out.println("---------------------");
}
}
```

Array list

When constructing an array list of strings, use

```
ArrayList<String> coll = new ArrayList<>();
```

an array list of integers would be

```
ArrayList<int> coll = new ArrayList<>();
```
[5]

Linear search:

Linear search is rarely used practically because other search algorithms such as the binary search algorithm and hash tables allow significantly faster searching comparison to Linear search.

Example :[6]

```
Input : arr[] = {10, 20, 80, 30, 60, 50,
                      110, 100, 130, 170}
          x = 110;
```

[5] Lecture slides, slides 7, week 20

[6] https://www.geeksforgeeks.org/linear-search/

```
Output : 6
Element x is present at index 6

Input : arr[] = {10, 20, 80, 30, 60, 50,
                      110, 100, 130, 170}
           x = 175;
Output : -1
Element x is not present in arr[].
```

Examples from lab:

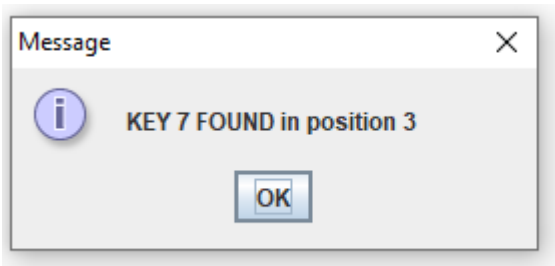
```java
package w15linear;
public class W15linear {
public static void main(String[] args) {

    int a[] ={5,7,1,13,24,16,7,8};
    int result;
    result=linearSearch(a,13,8); // result will get the value 3
    result=linearSearch(a,4,8);// result will get the value -1

} //main

public static int linearSearch (int[]data, int key, int sizeOfArray) {
    for(int i=0; i<sizeOfArray; i++) {
        if(data[i]==key)
            return i; // return position where found

    }//for end
    return -1;

}//public
}//public class end
```

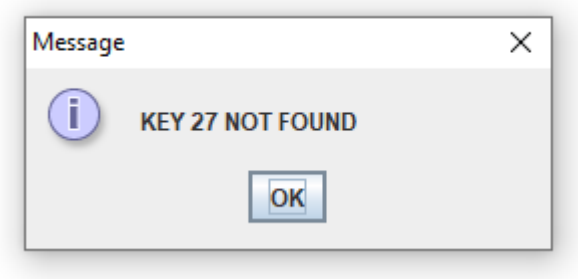
And another example:

We have been tested even numbers in this case. Look the result If I Put the number from list and outside list what is happened.

```
package testlinearsearchw16;
import javax.swing.*; // even numbers//
import java.util.*;
public class TestLinearSearchw16 {
public static void main(String[] args) {
        final int ARRAY_SIZE = 10;
        int [] a = {1,3,5,7,19,11,13,15,17,19};  // some test data to search
        int result; // result of this search
        int searchKey;
        searchKey = Integer.parseInt(JOptionPane.showInputDialog("Give me a number to search for"));
        result = linearSearch(a, searchKey, ARRAY_SIZE);
       if (result >= 0) // a valid array index
       JOptionPane.showMessageDialog(null,"KEY " + searchKey + " FOUND in position " + result);
        else
        JOptionPane.showMessageDialog(null,"KEY " + searchKey + " NOT FOUND");
        System.exit(0);
        }
        public static int linearSearch (int[] data, int key, int sizeOfArray)
        {
        for (int counter = 0; counter < sizeOfArray; counter++)
        {
        if (data[counter] == key)
        return counter; // return position where found
        }
        return -1; // if drop off end of array, its not there
}// end of main
}// end of public class
```

Result:





Get/set

Encapsulation,[7] is to make sure that "sensitive" data is hidden from users. To achieve this, you must: declare class variables/attributes as private and provide public get and set methods to access and update the value of a private variable.

\+ of encapsulation:

- better control of class attributes and methods
- class attributes can be made read-only (if you only use the get method), or write-only (if you only use the SET method)
- flexible: the programmer can change one part of the code without affecting other parts
- increased security of data.

The GET method returns the variable value, and the SET method sets the value.

[7] https://www.w3schools.com/java/java_encapsulation.asp

Example of use get and set in my project:

```
public  Flowerss (int inputquantity,String inputname, String inputcountry, String inputtype, double inputprice) {
 quantity = inputquantity;
 name = inputname;
 country = inputcountry;
 type = inputtype;
 price = inputprice;
}

public void setQuantity (int inputquantity) {
quantity = inputquantity;
}

public void setName (String inputname) {
name = inputname;
}

public void setCountry (String inputcountry) {
country = inputcountry;
}

public void setType (String inputtype) {
type = inputtype;
}

public void setPrice (double   inputprice) {
 price = inputprice;
}
```

Conclusion:

The project is made up for some methods, which have been used and explained during lecture. The aim of this project was to deep the knowledge and practice some methods which was done.