Day 2

Press Space for next page →

Table of contents

Hello World in Java

```
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello, world!");
  }
}
```

Java Basics

Data Type sizes

Type	Size (in bytes)
byte	1
short	2
int	4
long	8
float	4
double	8
boolean	1

Signed vs Unsigned Types in Java

Java has two types of signed integers: `int` and `long`. The difference is that the size of a `long` is twice as large as an `int`.

Range of values which can be be stored in a variable of a type

Calculation (unsigned integer)

Suppose size of the type is N bits. The total number of values that can be stored in a variable is 2^N.

`0000000` -> 8 times (lowest)

The value is 0.

`1 1 1 1 1 1 1` -> 8 times (highest)

= 2^8 - 1 = 256 - 1 = 255

Calculation (signed integer)

For 8 bits, there are 256 positions.

- 256/2 = 128 There is 0 128 <0> 127

The range is -128 to 127.

We had 8 bits. 2^(8) = 256 2^(8-1) = 128

-2^(8-1) to 2^(8-1) - 1 = -128 to 127.

For generic N bits,

- -(2^(N-1)) to 2^(N-1)-1

Types and size in Rust (aside)

Just for comparison, here are the sizes of some types in Rust:

Type	Size (in bytes)
u8	1
i8	1
u16	2
i16	2
u32	4
i32	4
u64	8

Unicode

Unicode is a way to represent characters in computers.

UTF-16 can have values from 2^0 to 2^16 - 1, which is 65,536. This means that it can represent 65,536 different characters.

All differect UTF encodings:

1. UTF-8: 7 bits per character (most common)
2. UTF-16: 16 bits per character (used by most modern systems)
3. UTF-32: 32 bits per character (used for Unicode code points)

Unicode encoding

```java
public class HelloWorld {
    public static void main(String[] args) {
        char ch = 0;
//        System.out.println((int)(Character.MAX_VALUE));
        for (; ch ⩽ Short.MAX_VALUE; ch++) {
            System.out.printf("%d = %c\n", (int)ch, ch);
        }
    }
}
```

Packages

- External Package names are in this format: `com.companyname.packagename` This is opposite the website Domain name convention.

For example, if your website is `oci.oracle.com`, then your package name would be `com.oracle.oci`.

- Internal Package names are in this format: `java.<package>`. For example, `java.util`, `java.lang`, `java.util.concurrent.atomic`.

You can think of packages as folders on a file system.

Importing packages

```java
// use scanner to read input from the console
import java.util.Scanner;

public class HelloWorld {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        // read a string from the console
        String s = sc.nextLine();
        System.out.printf("You entered: %s\n", s);
        // close the scanner
        sc.close();
    }
}
```

Methods to read diff data types

```
sc.next(); // read a string
sc.nextInt(); // read an integer
sc.nextLong(); // read a long
sc.nextFloat(); // read a float
sc.nextDouble(); // read a double
sc.nextBoolean(); // read a boolean

// read a character from console
// There is no method to read a character from console
// use next() and get the first character from it.
sc.next().charAt(0);
```

Trailing newlines

```java
import java.util.Scanner;

public class HelloWorld {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        // read a int from console
        int n = sc.nextInt();
        // **flush the extra newline at the end of the int**
        sc.nextLine();
        // read the next line
        String s = sc.nextLine();
        System.out.printf("Number is : ++%d++ String is ++%s++\n", n, s);
        // close the scanner
        sc.close();
    }
}
```

Arrays

```java
import java.util.Scanner;

public class HelloWorld {
    public static void main(String[] args) {
        System.out.print("Please enter size of arr: ");
        var sc = new Scanner(System.in);
        var k = sc.nextInt();

        var arr = new int[k];
        for (var i = 1; i <= k ; i++) {
            arr[i-1] = i;
        }
        // prints the length of the array
        System.out.println(arr.length);
        // prints the address of the array
        System.out.println(arr.toString());
        // prints the array
        for (var i : arr) {
            System.out.print(i + " ");
        }
        // close the scanner
        sc.close();
    }
}
```

Wrapper classes

- Java has primitive data types like `int`, `float`, `double`, etc.
- There are corresponding wrapper classes for each of these, such as `Integer`, `Float`, `Double`, etc.
- These classes provide additional functionality and methods that can be used with the primitive data types.
- For example, you can use the `Integer` class to create an object from a primitive integer value.

Wrapper classes

```
// Convert an int to an Integer
int i = 10;
Integer integer = Integer.valueOf(i);
// Convert an int to a String
String str = Integer.toString(i);
```

In C, we have `itoa` and `atoi`.

`itoa` is a function in C that converts an integer to a string. `atoi` is a function in C that converts a string to an integer.

Fibonacci series

```java
import java.util.HashMap;
import java.util.Scanner;

public class Sample {
    // create a static instance of DP map for memoization
    static HashMap<Integer, Integer> dp = new HashMap<>();
    static HashMap<Integer, Integer> testMxSize = new HashMap<>();

    static void fiboIterative(int count) {
        int f1 = 0, f2 = 1;
        for (int i = 0; i < count - 1; i++) {
            int temp = f1 + f2;
            f1 = f2;
            f2 = temp;
            System.out.println(f2);
        }
    }

    static int fiboRecursive(int count) {
        if (count == 0) return 0;
        if (count == 1) return 1;
        return fiboRecursive(count-1) + fiboRecursive(count-2);
    }

    static int fiboDP(int count) {
        if (count == 0) return 0;
        if (count == 1) return 1;
        // TODO: save and memoize the values of previously computed fiboDP
        if (dp.containsKey(count)) return dp.get(count);
        var k = fiboDP(count-1) + fiboDP(count-2);
        dp.put(count, k);
        return k;
    }

    static void validateMapMaxSize() {
```

Seive of Eratosthenes

```
public class Sample {

    public static void main(String[] args) {
        // Upper limit for the primes we want to check for.
        int maxVal = 30;

        // We are running of Seive of Erathnosis to cut out non-primes.
        boolean[] isPrime = new boolean[maxVal+1];

        // Set all numbers by default as prime.
        for (int num = 2; num <= maxVal; num++) {
            isPrime[num] = true;
        }

        for (int num = 2; num <= maxVal; num++) {
            if (!isPrime[num]) continue;
            for (int multiple = num*2;
                 multiple <= maxVal; multiple += num) {
                isPrime[multiple] = false;
            }
        }

        // Printing primes
        for (int num = 2; num <= maxVal; num++) {
            if (isPrime[num]) System.out.println(num);
        }
    }
}
```

Data Structure time complexity Parameters

- **Insertion at the ends**
- **Insertion at any position**
- **Deletion at the ends**
- **Deletion at any position**
- **Search / Cheeck for existence**
- **Order between elements**
- **Find by order**
- **Get order of an element**

ArrayLists in Java

- ArrayLists are a part of the Java Collections Framework (`java.util` package).
- Differences from arrays **Dynamic size** (can grow or shrink)

Using ArrayList instead of arrays

```
import java.util.ArrayList;
import java.util.Collections;

public class Sample {

    public static void main(String[] args) {
        // Upper limit for the primes we want to check for.
        int maxVal = 30;

        // create a static sized (31) length arraylist of boolean with default value of true.
        // We dont need to initialize the values if we use arraylist.
        var isPrime = new ArrayList<Boolean>(Collections.nCopies(maxVal+1, true));

        // c++ equivalent.
        // auto x = vector<bool>(maxVal+1, true);

        for (int num = 2; num <= maxVal; num++) {
            if (!isPrime.get(num)) continue;
            for (int multiple = num*2;
                 multiple <= maxVal; multiple += num) {
                isPrime.set(multiple, false);
            }
        }

        // Printing primes
        for (int num = 2; num <= maxVal; num++) {
            if (isPrime.get(num)) System.out.println(num);
        }

    }
}
```

Grid questions.

Reference: https://cses.fi/problemset/task/1192

You are given a map of a building, and your task is to count the number of its rooms. The size of the map is n*m squares, and each square is either floor or wall. You can walk left, right, up, and down through the floor squares.

Please check github for code

https://github.com/nascarsayan/java-core-lpu/blob/main/slides/md/day2_java.md#please-check-github-for-code

```java
import java.util.*;

public class ConnectedComponents {
    enum Cell {
        EMPTY,      // represents '.' (was 0)
        VISITED,    // for BFS marking (was 1)
        WALL        // represents '#' (was 2)
    }

    static class Pair {
        int first, second;

        Pair(int first, int second) {
            this.first = first;
            this.second = second;
        }
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        int nr = sc.nextInt();
```