JAVA

LEARNING

Ayushi Dixit | NOTES | 25-09-2024

Primitives :

- The most basic fundamental data types in JAVA.
- The primitive types represent single values—not complex objects.
- Java defines eight primitive types of data**: byte, short, int, long, char, float, double, and boolean.**
- sin( ), cos( ), and sqrt( ), return double values.

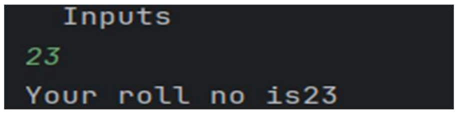
```
public class primitives {
    public static void main(String[] args) {
        int marks = 98;
        char section = 'A';
        byte age = 100; // 8 bit
        short roll_no = 7484; //16 bit type
        long phone_no = 987654321; //64 bit type
        float weight = 10.56f; //real numbers 32 bits
        double size_of_celestrial_body = 34567822875428866442143677 22.645; //64
bits
        boolean check = true;

    }
}
```

In some cases float throws error (Floating point error) when we use decimal in it so, we use 'f' for float to represent decimal values in Float.
IMP:

- String is not in Primitives
- In JAVA strings are written in (" ") while char in (' ')

INPUTS

```
Scanner input = new Scanner(System.in);
int roll_no = input.nextInt();
System.out.println("Your roll no is" + roll_no);
```



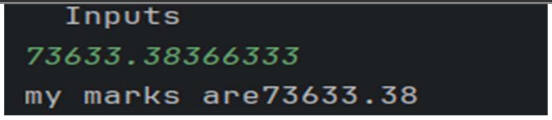
```
   Inputs
 23
 Your roll no is23
```

IMP:

- print() :- the next output will be printed on the same line, immediately.
- println() :- adds a new line character (\n) at the end. The next output will be printed on a new line.

Floating point error:-

```
float marks = input.nextFloat();
System.out.println("my marks are" + marks);
```



```
   Inputs
 73633.38366333
 my marks are73633.38
```

The digits got round off, this is the floating point error.

Type conversion & casting

Assigning a value of one type to a variable of another type.
Java performs the conversion automatically if the types are compatible
 e.g. int value can be assigned to a float var BUT double can't be assigned to byte.

An automatic conversion;

- Types are compatible
- Destination type is larger than the source type
- No automatic conversion of num to char or Boolean.

```
int num = 'A';
System.out.println(num);
```

```
65
```

here, we r giving a string in integer and it results in an integer
65 is the ASCII value is A.
Java follows the Unicode rules, hence any language can be used.

TRUNCATION

When floating point value is assigned to an integer type, the fractional component is lost.

```
float a = 43.5f;
int s = (int) (a);
System.out.println(s);
```

```
43
```

Casting 257 to a byte variable.

As the range of byte is 256, if we assign a larger value it results as [257 % 256 == 1]

```
int a = 259;
byte b = (byte) (a);
System.out.println(b);
```

```
3
```

```
byte x = 60;
int y = x;
System.out.println(y);
```

```
60
```

here a byte x is assigned to an int variable, hence destination type is larger than a source type. But the reverse is not true.

Here comes the role of CASTING

Assigning an int to a byte variable is narrowing conversion.

```
int a = 23;
int b = 34;
```

```
byte c = (byte) ((byte) (a) + (byte) (b));
System.out.println(c);
```

```
57
```

Automatic Type Promotion in Expressions

Java automatically promotes each byte, short, or char operand to int when evaluating an expression.

```
byte a = 40;
byte b = 30;
byte c = 100;
int d = a *b /c ;
System.out.println(d);
```

```
12
```

Here a * b easily exceeds the range of a byte operand i.e. 256, here java automatically handled the expression and a * b is performed using integers.
This can also sometimes cause a problem e.g.

```
byte b = 20;
b = b*2;
System.out.println(b);
```

```
java: incompatible types: possible lossy conversion from int to byte
```

the operands were automatically promoted to int, hence the result of the expression is now of type int, which cannot be assigned to a byte without the use of a cast.
Casting:

```
byte b = 20;
b = (byte) (b*2);
System.out.println(b);
```

```
40
```

```
import java.io.File;
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello and welcome!");
        Scanner input = new Scanner(System.in);
    }
}
```

Output:

```
Hello and welcome!
```

SO, basically

- Scanner here is the CLASS in java.util package that allows us to take the input.
- System.in takes the input from the keyboard. We can change it to the type you want e.g. File
- Input = Variable
- The public means the class can be accessed from anywhere in the program.
- void means the method doesn't return anything.

```java
import java.io.File;
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println(input.nextInt());
    }
}
```

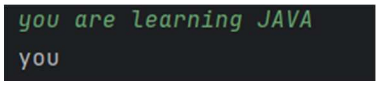
Output:

```
Input 456
456
```

Here, System.out.println(input.nextInt()); reads an integer value from the user input and then prints the value to the console

To input String

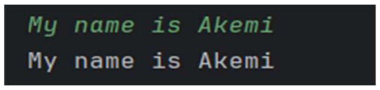
```java
System.out.println(input.next());
```



```
you are learning JAVA
you
```

Scanner uses white space as a delimiter so this will print only one word from the string, To get the whole line printed use

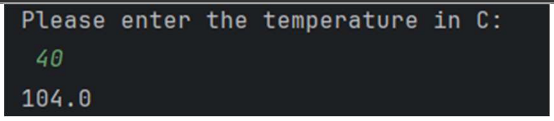
```java
System.out.println(input.nextLine());
```



```
My name is Akemi
My name is Akemi
```

A Simple JAVA program:

```java
import java.util.Scanner;

public class temperature {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.println("Please enter the temperature in C:");

        float C = in.nextFloat();
        float F = (C * 9/5) + 32;
        System.out.println(F);
    }
}
```



```
Please enter the temperature in C:
 40
104.0
```