To extract:

$ sudo dpkg -i name\_of\_the\_file

$ sudo update-alternatives --install /usr/bin/java java /usr/lib/jvm/jdk-14/bin/java 1

$ sudo update-alternatives --install /usr/bin/javac javac /usr/lib/jvm/jdk-14/bin/javac 1

$ java --version

$ sudo update-alternatives --config java

Installing extensions in vscode

1. Language support for java by Redhat

2. Java debug by bin Deng

Alt+C (to compile)

Alt+R (to run)

/\*\* A program to display the message

\* "Hello World!" on standard output.

\*/

public class HelloWorld {

public static void main(String[] args) {

System.out.println("Hello World!");

}

} // end of class HelloWorld

The command that actually displays the message is:

System.out.println("Hello World!");

The word "public" in the first line of main() means that this routine can be called from outside the program.

A Program looks like:

**optional-package-declaration**

**optional-imports**

public class **program-name** {

**optional-variable-declarations-and-subroutines**

public static void main(String[] args) {

**statements**

}

**optional-variable-declarations-and-subroutines**

}

The program-name in the line that begins "public class" is the name of the program, as well as the name of the class.

Identifiers can be used to name classes, variables, and subroutines. It must begin with a letter or underscore and must consist entirely of letters, digits, and underscores.

Java is actually pretty liberal about what counts as a letter or a digit. Java uses the Unicode character set, which includes thousands of characters from many different languages and different alphabets, and many of these characters count as letters or digits.

An assignment statement takes the form:

**variable** = **expression**;

where **expression** represents anything that refers to or computes a data value.

A literal is something that you can type in a program to represent a value. It is a kind of name for a constant value. A tab is represented as '\t', a carriage return as '\r', a linefeed as '\n', the single quote character as '\'', and the backslash itself as '\\'. Note that even though you type two characters between the quotes in '\t'

A literal is something that you can type in a program to represent a value. It is a kind of name for a constant value.

First of all, real numbers can be represented in an exponential form such as 1.3e12 or 12.3737e-108.

To make a literal of type float, you have to append an "F" or "f" to the end of the number.  For long, use ‘L’ or ‘l’ .

Octal -> starts with ‘0’

Hexa -> starts with ‘0x’ or ‘0X’

Binary -> with ‘0b’ or ‘0B’

As a final complication, numeric literals can include the underscore character ("\_"), which can be used to separate groups of digits. For example, the integer constant for two billion could be written 2\_000\_000\_000, which is a good deal easier to decipher than 2000000000.

For the type boolean, there are precisely two literals: true and false.

To find the range of primitive types:

int myMinIntValue = Integer.MIN\_VALUE;

int myMaxIntValue = Integer.MAX\_VALUE;

if we use something like:

System.out.println(“Maximum Value = “ + myMinIntValue);

Here, the myMinIntValue will be converted into string and displayed

Wrapper classes: Integer for int to perform operation on integer

Casting in Java: In parentheses, the computer converts the data into int

B = 60;

byte a = (byte)(b/2); [\\Right](../../../../..//Right) byte a = (b/2);

When precise calculation is required for fractional numbers, BigDecimal class can be used.

To use Unicode,

char myOOda = ‘\u0AF4’;

In Strings,

String myString = “This is amazing \u00A9 haha”;

Strings in Java are Immutable (it cannot be changed)

If {

}else if{

}else{

}

In coding conventions for searching algorithms, -1 means value not found or invalid value.

DiffMerge is a program that will help you to visually compare and merge files on any operating system.

Reading Inputs:

Scanner inputObject = new Scanner(System.in);

boolean hasNextInt = inputObject.hasNextInt();

if (hasNextInt) {

int anyNumber = inputObject.nextInt();

} else {

System.out.println("Invalid Number!!");

}

inputObject.nextLine();

Making Constructors:

public Class\_name() {

}

public Class\_name(String name, double balance) {

this.name = name;

this.balance = balance

}

Calling Constructor from a constructor:

public Class\_name(){

this(“Sharan”, 280); //this has to be the very first line of the constructor

}

To compile:

javac \*.java

To run:

java main\_function\_file\_name