Date ___/___/_____ Page No. _____

Java Basics

File Name : Demo.java Class Name : Demo
good practice to use initial character as capital

☆ Basic Terminologies :

1. public → used so that we can use class from anywhere
2. functions → collection of code that we can use again and again. (aka methods)
3. void → specifies that a method shall not have a return value
4. String [] args → means an array of sequence of characters ("strings") that are passed to the main function

☆ Note :

- after compiling .class file is always stored in current location where you are in
- we can use -d (destination) option while compiling and specifying the path

javac -d <path> Demo.java

5. echo $PATH → every command looks for this location before executing [environment variables]

☆ Note :

class name & file name should be same, but if we do not want to make class name as file name then it must be ~~private~~ not be public | class Divide |

ABHINAV

Date __/__/____ Page No ____

→ package com.abc.file1

- basically means we are inside the folder

com
└ abc
  └ file1
└ def
└ xyz
└ pqr

```
System.out.println("Hello");
```

System → class
out → var
println → fn/method

This means print the output or Standard output stream

println → adds new line
print → doesn't add new line

```
Scanner input = new Scanner(System.in);
```

Scanner → class that allows us to take input
new → creating object
System.in → take input from standard input (here, keyboard)

* Primitive
means that any datatype cannot be further
Integer, Character are primitive datatypes

ABHINAV

Date ___/___/_____ Page No. _____

* Data types in Java:

```
int rollno = 64;
char letter = 'r';
float marks = 64.65f;
double dnum = 45.6754;
long lnum = 24653422L;
boolean check = true;
```

why the letters

Note:
All decimals values are stored as a double datatype, therefore if we want to store in float or any other datatype, then we'll have to specify it by writting 'f' or 'F' or 'L' in the end

→ **Integer** → wrapper class → provides additional functionalities
↓
converts primitive datatype to object

```
int a = 10;
```
identifier — literal

- **Literals:** Java literals are syntactic representations of boolean, character, numeric or string data. Here, 10 is an integer literal.
- **Identifiers:** Identifiers are the names of variables, methods, classes, packages and interfaces.

ABHINAV

Date ___/___/_____ Page No. _____

Note:

- int a = 3600_467_890;
  ↳ In this case the value of the integer will be 3600467890.

- 564.12345678 —rounds off→ 564.12345
  ↳ if we have give float a very big value, then it rounds it off, which gives floating point error

* Type casting & Type Conversion:

(1.) Widening or Automatic Type Conversion:
→ two datatypes are automatically converted
→ happens when we assign value of smaller datatype to bigger datatype and two datatype must be compatible

byte → short → int → long → float → double

eg →

```
int   i = 100 ;  → 100
long  l = i   ;  → 100
float f = l   ;  → 100.0
```

(2.) Narrowing or Explicit Conversion:
→ This happens when we want to assign a value of larger data type to a smaller data type
we perform explicit type casting or narrowing

double → float → long → int → short → byte

ABHINAV

Date __/__/____ Page No. ____

eg →
double d = 100.04; → 100.04
long l = (long)d; → 100
int i = (int)l; → 100

(3.) Automatic type promotion in Expressions :
→ while evaluating expressions, the intermediate value may exceed the range of operands and hence the expression value will be promoted.
→ some conditions of type promotion are :
(1.) Java automatically promotes each byte, short, char to int when evaluating an expression.
(2.) Long, float or double, the whole expression is promoted to long, float or double

eg : After solving expression
(y * b) + (i / c) - (d * s)
we get → (float) + (int) - (double)
(converted to double)
= double

(4.) Explicit type casting in expression :
→ If we want to store large value of into small data type.
eg : byte b = 50;
b = (byte)(b * 2)
↳ type casting int to byte

ABHINAV

Date __/__/__ Page No.:

* If-else syntax :

```
if (condition) {
    // block of code
} else {
    // block of code.
}
```

* For loop syntax :

```
for (statement 1;
statement 2; statement 3)
{ // block of code }
```