Page No.
Date

* Basics of Java Programming *

* Why we are here?

- Make Learning online.
- New technology in industry.
- know about new techie mind and what they thing about coding, programming & language.
- Make you easy about software company.

* A brief history of Java.

- "Java, whose original name was Oak, was developed as a part of the Green project at Sun. It was started in December '90 by Patrick Naughton, Mike Sheridan and James Gosling and was chartered to spend time trying to figure out what would be the "next wave" of computing and how we might catch it.

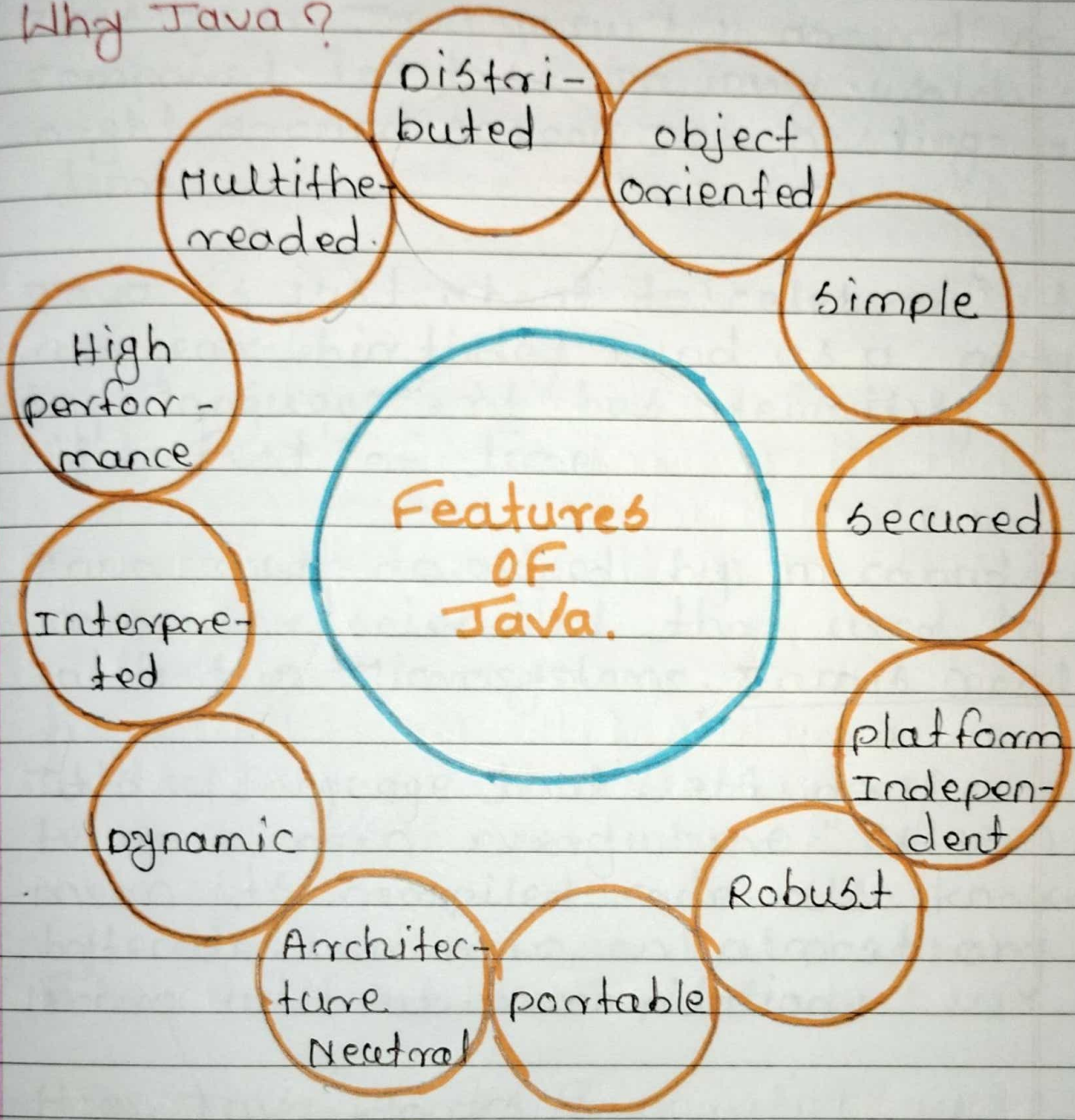
Page No.
Date

When Java ?

Version	Date.
JDK 1.0	January 23, 1996
JDK 1.1	February 19, 1997
J2SE 1.2	December 8, 1998
J2SE 1.3	May, 8, 2000
J2SE 1.4	February 6, 2002
J2SE 5.0	September 30, 2004
Java SE 6	December 11 2006
Java SE 7	July 28, 2011
Java SE 8	March 18, 2014
Java SE 9	September 21, 2017
Java SE 10	March 20, 2018
Java SE 11	September 25, 2018
Java SE 12	March 19, 2019
Java SE 13	September 17, 2019
Java SE 14	March 17, 2020

Page No.
Date

Why Java ?



Page No.
Date

- This year Java grew by around 60% compared to last January, which was right around 62,000 Job postings at the time.

- Java is just about to celebrate its 24-year birthday, and as a programming language, it has definitely stood the test of time.

- Java was developed by a canadian computer scientist that used to work with Sun Microsystems, James Gosling

- It's a language that lets developers "write once, run everywhere;" (WORA), which mean its compiled code, also known as bytecode, can run on almost any platform without recompilation.

How Java Works ?

- Java's platform independence is achieved by the use of the Java virtual Machine (JVM)

- A java program consists of one or more files with a .java extension.

Page No.
Date

- When a java program is compiled the .java files are fed to a compiler which produces a .class file for each .java file
- The .class file contains java bytecode
- Bytecode is like machine language, but it is intended for the Java virtual Machine not a specific chip such as a Pentium or PowerPC chip
- To run a java program the bytecode in a .class file is fed to an interpreter which converts the byte code to machine code.
- Some people refer to the interpreter as "The Java virtual Machine" (JVM)
- The interpreter is platform specific because it takes the platform independent bytecode and produces machine language instructions for a particular chip.
- So a Java program could be run an any type of computer that has a JVM written for it.
  - PC, Mac, Unix, Linux, BeaOS, Spare

Page No.
Date

So What!

- The platform independence of Java may be a huge marketing tool, but is actually of little use to people learning object oriented Programming and Abstract Data Types.

- Java is a "pure" object oriented language

  - encapsulation, inheritance, and polymorphism.

  - all code must be contained in a class

Hello World.java Program

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

Where Java?

- Notes:- year after year, Java is the #1 most searched programming language.

Page No.
Date

* Tools and Technology used in class:-
  - JDK (1.8 or above version)
  - Notepad
  - Eclipse Mars
  - Netbeans
* How to make Java as a hobby?
  - Start from Simple
  - co-Related it to day to day life.
  - once start, Finish any how!
  - Momentum slow but steady.

Basic features

- Data Types
  - primitives
  - classes / objects
- Expressions and operators
- control structures
- Arrays
- Methods
- Programming for correctness
  - Pre and post conditions
  - assertions

* Identifiers in Java :-

- Letters, digits, _, and $ (don't use $. can confuse the runtime system)
- Start with letter, _, or $
- By convention:

1) Start with a letter

2) variables and method names, lowercase with internal words capitalized e.g. honkingBigVariableName.

3) constants all caps with _ between internal words e.g. ANOTHER_HONKING_BIG_INDENTIFIER

4) classes start with capital letter, internal words capitalized, all other lowercase e.g. HonkingLongClassName.

* Data Types :-

- Primitive Data Types :-

  - byte short int long float double boolean char

  - stick with int for integers, double for real numbers.

- Classes and Objects :-

  - pre defined or user defined data types consisting of constructors, methods, and fields. (constants and fields (variables) which may be primitive or objects.)

Page No.
Date

Java Primitive Data Types:-

Data Type	characteristic	Range
byte	8 bit signed integer	-128 to 127
short	16 bit signed integer	-32768 to 32767
int	32 bit signed integer	-2,147,483,648 to 2,147,483,647
long	64 bit signed integer	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	32 bit floating point number	±1.4E-45 to ±3.4028235E+38
double	64 bit floating point number	±4.9E-324 to ±1.79769313486231570E+308
boolean	true or false	NA, note Java booleans cannot be converted to or from other types
char	16 bit, unicode	unicode character, \u0000 to \uffff can mix with integer types.

Page No.
Date

What are classes and objects?

- class is synonymous with data type
- object is like a variable
  - The data type of the object is some class

    Referred to as an instance of a class
- classes contain :-
  - The implementation details of the data type
  - and the interface for programmers who just want to use the data type.
- objects are complex variables
  - usually multiple pieces of internal data
  - various behaviors carried out via methods.

Page No.
Date

Program :-

```
class calculation {// class
   int a = 10;
   int b = 20;
   void add() {
   int add = a+b;
    System.out.println("Addition of two nu-
                        mber" + add);
  }
public static void main(String args[]){
  calculation obj = new calculation(); //
   object creation -> instance of class

  System.out.println("value of 1st number:"
                                 +obj.a);
   obj.add();
  }
}
```

output :- value of 1st number : 10
Addition of two number: 30

Creating and Using Objects :-

- Declaration - Data Type identifier
  Rectangle r1;
- Creation - new operator and specified constructor
  r1 = new Rectangle ();
  Rectangle r2 = new Rectangle ();
- Behavior - via the dot operator
  r2.setSize (10, 20);
  String s2 = r2.toString ();

Built in Classes :-

- Java has a large built in library of classes with lots of classes with lots of useful methods
- ones you should become familiar with quickly.
- String
- Math
- Integer, character, Double.

- Systems
- File
- Arrays
- object
- Scanner
- Random
- Look at the Java API Page

Import :-

- Import is a reserved word
- packages and classes can be imported to another class
- Does not actually import the code (unlike the c++ include preprocessor command)
- Statement outside the class block

```
import java.util.ArrayList;
import java.awt.Rectangle;
public class Foo {
        // code for class Foo
}
```

- can include a whole package

```
import java.util.*;
```

- or list a given class
  - import java.util.Random;
- Instructs the compiler to look in the package for types it can't find defined locally
- The java.lang.* package is automatically imported to all other classes.
- Not required to import classes that are part of the same project in Eclipse.

The String class:-

- String is a standard Java class
  - a whole host of behaviors via methods
- also special (because it used so much)
  - String literals exist (no other class has literals)

```
String name = "Mike D.";
```

  - String concatenation through the + operator

```
String firstName = "Mike";
String lastName = "Scott";
String wholeName = firstName +
                   lastName
```

- Any primitive or object on other side of + operator from a string automatically converted to string.

★ Standard output :-

- To print to standard output the use

```
System.out.print(expression); // no newline
System.out.println(expression); // newline
System.out.println(); // just a newline
```

common idiom is to build up expression to be printed out

```
System.out.println("x is:" + x + " y is:" + y);
```

★ Constants :-

- Literal constants - " the way you specify values that are not computed and recomputed, but remain well, constant for the life of a program."

- true, false, null, 'c', "c++", 12, -12, 12.12345

- Named constants

- use the keyword final to specify a constant.

- Scope may be local to be a method or to a class

- By convention any numerical constant besides -1, 0, 1 or 2 requires a named constant

```
final int NUM_SECTIONS = 8;
```

* Program code:-

```
class DemoString{
    final static int age = 10;
public static void main (String args[])

    String firstName = "Tom";
    String lastName = "Jerry";

// age = 20; -> it will create error if we try this to change, due to constant

    String wholeName = firstName + lastName;
    System.out.print ("Whole Name:");
    System.out.println (wholeName);
}
}
```

Output:- Whole Name: TomJerry

Page No.
Date

* Operators :-

- Basic Assignment : =
- Arithmetic Operators : +, -, *, /, % (remainder)
  - integer, floating point, and mixed arithmetic and expressions.
- Assignment operators : +=, -=, *=, /=, %=
- increment and decrement operators : ++,

* Expressions :-

- Expressions are evaluated based on the precedence of operators
- Java will automatically convert numerical primitive data types but results are sometimes surprising.
  - take care when mixing integer and floating point numbers in expressions.
- The meaning of an operator is determined by its operands.

* Casting :-

- Casting is the temporary conversion of a variable from its original data type to some other data type.

- Like being cast for a part in a play or movie.

- With primitive data types if a cast is necessary from a ~~class~~ less inclusive data type to a more inclusive data type it is done automatically.

```
int x = 5;
double a = 3.5;
double b = a * x + a / x;
double c = x / 2;
```

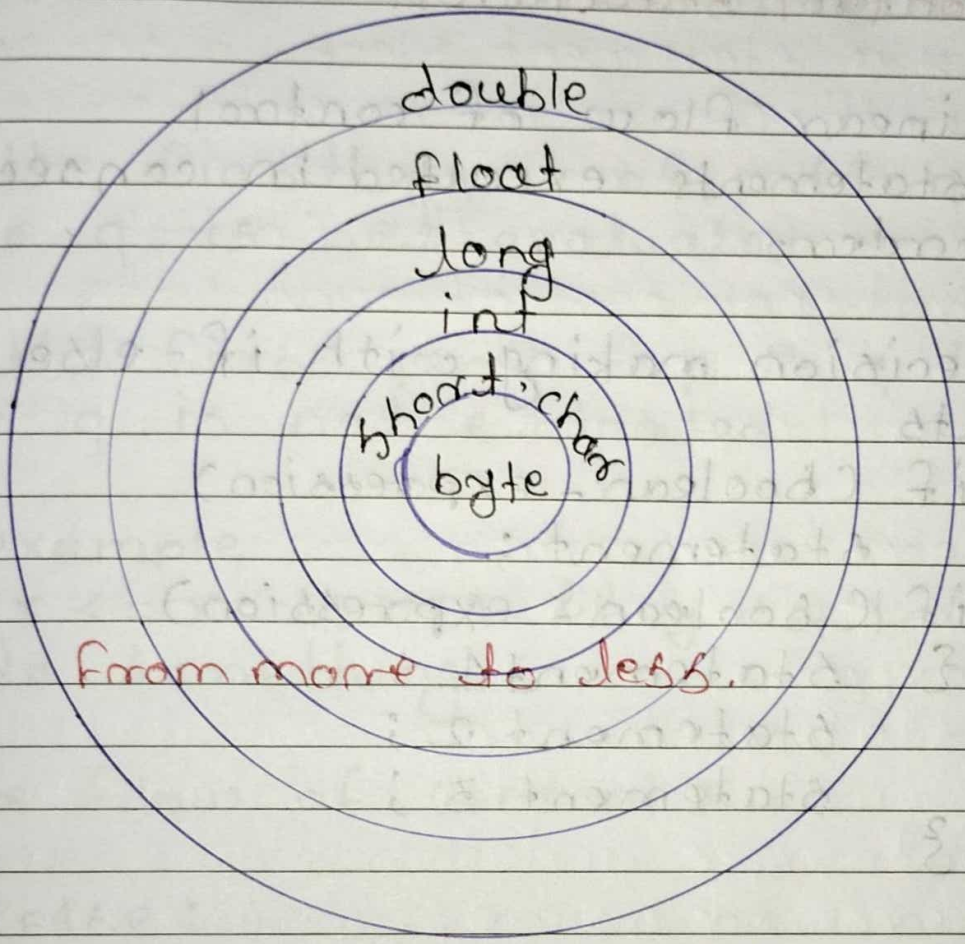
- if a cast is necessary from a more inclusive to a less inclusive data type the class must be done explicitly by the programmer.

- failure to do so results in a compile error.

```
double a = 3.5, b = 2.7;
int y = (int) a / (int) b;
y = (int) (a / b);
y = (int) a / b ; // Syntax error
```

Page No.
Date

Primitive casting.



- outer ring is most inclusive data type. Inner ring is least inclusive.
- In expressions variables and sub expressions of less inclusive data types are automatically cast to more inclusive.
- If trying to place expression that is more inclusive into variable into variable that is less inclusive, explicit cast must be performed.

Page No.
Date

Control structures:-

- linear flow of control
  - statements executed in consecutive order.

- Decision making with if-else statements

```
if (boolean-expression)
    statement;
if (boolean-expression)
{   statement1;
    statement2;
    statement3;
}
```

- A single statement could be replaced by a statement block, braces with 0 or more statements inside.

Boolean Expressions:-

- Boolean expressions evaluate to true or false.
- Relational operators : >, >=, <, <=, ==, !=
- Logical operators : &&, ||, !