Agenda

- Revision
- main() variations
- Console input and output
- Language Fundamentals
    - Naming Conventions
    - Comments
    - Keywords
    - Datatypes
- Widening and Narrowing
- Literals
- Variable
- Operators
- Wrapper classes
- Boxing & UnBoxing
- Command Line Arguments
- Control Statements
- Java method

Shortcut for STS

- open the sts.desktop in text editior
- copy the path of STS application from the directory where you have kept
- edit the sts.desktop file where we need to add your STS path inside the file.
- change Icon and Exec path inside the sts.desktop file.
- copy this sts.desktop file on your desktop and rightclick on it, select Allow launching.
- your STS shortcut is now ready to use.

main() variations (Demo01)

- We can write multiple main inside multiple classes
- we can also write main in multiple .java files
- java creates a .class file for every class that we write
- if we define mutliple classes in the single .java file , still their will be .class file for every class that you have defined inside that .java file.
- we can write the main() in every class, and also we can execute it individually

```
// error -> Main method not found
public static void main(String args) {
        System.out.println("main() Program03");
    }

  //error -> main is not static
  public void main(String[] args) {
        System.out.println("main() Program03");
    }
```

```
  // error -> Main method not found
  public static void Main(String[] args) {
        System.out.println("main() Program03");
    }

  //correct way
  public static void main(String[] args) {
        System.out.println("main() Program03");
    }
```

- we can also overload the main method

```
  public static void main(String args) {
        System.out.println("main() with string data");
    }

    public static void main() {
        System.out.println("main() without parameters");
    }

    public static void main(String[] args) {
        System.out.println("main() Program03");
        main("sunbeam");
        main();
```

Console input and output (Demo02)

- Java have provided multiple ways to deal with input and output from the console.
- the most popular ways are
- 1. using Console class
    - it is inside the package called as java.io

```
// Will  not work with STS (with any ide)
// It should be executed from the command line
public static void main(String[] args) {
        String name;
        String mobile;

        Console console = System.console();

        System.out.print("Enter name = ");
        name = console.readLine();

        System.out.print("Enter mobile = ");
        mobile = console.readLine();
```

```
        System.out.println("Name = "+name);
        System.out.println("Mobile = "+mobile);

    }
```

- 2. using Scanner class
  - it is from the package java.util

```
    public static void main(String[] args) {
        int rollno;
        String name;
        double marks;

    // creating the object of scanner class
    // we need an object of Inputstream which we can fetch from System
class.
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter rollno = ");
        rollno = scanner.nextInt();

        System.out.print("Enter name = ");
        name = scanner.next();

        System.out.print("Enter marks = ");
        marks = scanner.nextDouble();


        System.out.println("Roll No = "+rollno);
        System.out.println("Name = "+name);
        System.out.println("Marks = "+marks);

    }
```

Language Fundamentals

Naming Conventions

- 1. camelCase
  - First letter of every word except first world is kept in upppercase
  - eg -> totalSalary;
  - this naming convention is used for
    - 1. local variables
    - 2. class fields
    - 2. method names
    - 3. method parameters
- 2. PascalCase

    - First letter of every word is kept capital
    - this naming convention is used for
        - 1. class
        - 2. initerface
        - 3. enum

```
class Employee{

}
class StudentAttendance{

}
```

- 3. naming convention for packages
    - names for all the pacakges should be in lowercase.
    - eg->
        - java.lang
        - java.io
        - java.util
- 4. Contants
    - contant fields should be kept in ALL CAPS

```
public final PI = 3.14;
```

Comments

```
// single line comment
//int num1 = 10;


// Multiline Commnets
  /*
     * void f1(){
   *  sysout("Inside f1");
   *  }
    */

// Block Comments
/*
      void f1(){
     sysout("Inside f1");
     }
*/
```

Keywords

- these are the reseverd words that have special meaning for them.
- eg->
    - abstract
    - boolean
    - char
    - double
    - enum
    - float
    - goto
    - interface
    - long
    - new
    - ..

Datatypes

- Data types defines 3 things
    - 1. Nature
    - 2. Memory
    - 3. Operations
- 1. Primitive Datatype (Value types)
    - Boolean -> boolean (1 bit) -> true or false
    - Character -> char (2 bytes)
    - Integral -> byte(1 byte),short(2 bytes),int(4 bytes),long (8 bytes)
    - Floating point -> float(4 bytes), double(8 bytes)
- 2. Non Primitive Datatypes (Reference types)
    - class
    - enum
    - interface
    - Array

Literals

- six types of literals in java
    - 1. Integer Literals
    - 2. Floating Point Literals
    - 3. Character Literals
    - 4. String Literals
    - 5. Boolean literals
    - 6. null Literal

```
int num1 = 30; // Integral literals

double num2 = 123.456; // floating point literals
```

```
char a = 'A'; // character literal

String name = "sunbeam" /// String literal

boolean status =  true or false // Boolean literals

Scanner sc = null; // null literal
```

Widening and Narrowing (Demo04 -> Program01)

- the process of converting narrower type of data into wider type is called as widening
- the process of converting wider type of data into narrow type in called as narrowing
- at the time of marrowing explicit typecasting is mandatory

Variable

- varaible is a container that is used to store the data/value inside it.
- variable is used to identity the address inside the memory
- to declare a variable we use below syntax
    - datatype variable_name(identifier) = value;
- variable can be of value type or reference type
- if a variable is declared inside a class it is called as field
- if we create a variable of a class it is called as a reference
- variables can be initialized at the time of declaration or ot can be assigned the values later.

Operators

- In java we have below defined category of operators
    1. Arithmetic Operators : +,-,*,/,etc
    2. Assignment Operator : =, +=,-=,etc
    3. Comparision Operators : ==,<=,>=,etc
    4. Logical Operators : &&,||, !,etc
    5. BitWise Operator : &,|,~,>>,<<,etc
    6. Misc Operators : Ternary operator(? 😃
    7. dot operator : Classname.membername, object.membername.

Wrapper classes

- All primitive types are not classes.
- java have provided classes for every primitive types
- these classes are called as wrapper classes

```
- Object
    - Boolean
    - Character
```

```
    - Number
        - Byte
        - Short
        - Integer
        - Long
        - Float
        - Double
```

- why to use wrapper classes
    - for conversion between value type to reference type.
    - to get the size and max and min value of the primitive datatypes.
    - to use the helper/utility methods
    - java collection stores the data of reference type only.

Boxing and Unboxing (Demo04-> Program02, Program03)

- Conversion of value type into reference type is called as boxing
- If the conversion of value type into reference type is directly possible without of any helper methods from the wrapper classes it is called as auto boxing
- Conversion of reference type into value type is called as unboxing
- If the conversion of refrence type into value type is directly possible without of any helper methods from the wrapper classes it is called as auto unbboxing

Command Line Arguments (Demo04-> Program04)

- write the code into STS and execute the code from the terminal by going inside the bin directory
- when executing pass the command line arguments
    - java Program04 10 20
- We can also pass the command line arguments into STS
- right click on the Program01 -> select Run as -> Run Configuration
- in the arguments tab provide the program arguments and select run

Control Flow Statements (Demo05)

- In java all the statements are executed sequentially
- If we want to control the flow of these statements we use Flow Control statements
- We have below category of Flow Control Statements
    - 1. Decision Making Satements
        - if statement
        - else if statement
        - switch statement
    - 2. Loop Statements
        - do..while();

        - while()
        - for ()
        - for-each
    - 3. Jump Statement
        - break
        - continue

- in java we can use string literals in case statement

Java Method

- the functions that we write in C/C++ are called as Methods in Java
- for method name camelCase naming Convention is used.
- a method can be parameterless or paramterized
- a method can return something or it can be void
- a method if itis returning something it can be a value type or a reference type
- a method can be static or non static
- static methods need to called on classname using . operator
- non static methods need to called on objects.

LabWork

- Read the help file Max 30 mins
- Solve the assignments
- If required work on the classwork demos

Prerequisite

- Class
- Object
- Types of member functions
- Constructor
- this pointer
- new -> dynamic object (diagram)