28/10/22 ⑥ Class Identifiers, Reserved Keyword, Data types in JAVA :-

Today's topic of discussion

① oops (basic introduction)
② identifiers / Variables
a. rules to write an identifier.

③ Reserved words.
④ Data types and it's chart.

① Object orientation Principles :- (opps)

→ it stands for object oreintation principles.
object → real time instance

→ Every object in realtime will have 2 parts.
① what it has?
② what it does.

Eg Car :-

① what it has?
brandname
no. of wheels
model
Speed

② what it does?
move
accelerate
brake.

Java Code → object representation

```
Class Car
{
    // Has part of an object is represented as a
                                        "Variable".
String  brandName
 int    noofWheels
String  model
        Speed
    // Does part of an object is represented as
                              Through "methods".
}
```

→ terminate statements with, Semi Colon;

Java Code

object representation to Express object we are Class.

```
Class Car
{
  // Has part of an object is represented as
    Variable.
  String  brandName;
  int     noofwheels;

  // Does part of an object is represented
    through Methods.
  Public Void move()
  {
    // logic of Moving a Vechicle
  }
   Public Void accelerate()
   {
    // lorgic of accelerating a Vehicle
   }
}
```

① object :- what we see physically.

Eg:- ② student → object

```
Class Student
{
    // Has part (variables)
String name;                      ] — Has Part
int id.;
float height;

    // Does part (methods)
    Public Void study(){
      // logic of studying.          ] → does part
    }
    Public Void playing(){
      // logic of playing.
    }
}
```

② object :- object is a member (also called as instance) of a Java class. Each object has an identity, a behavior and a state.

→ the state of a object is stored in fields (variable)

→ while (methods (functions) display's the objects behavior.

② ~~variable~~ identifier :- 29/10/22

→ identifier is a name in java program

→ it Can be a class name, method name, Variable name and lable name.

```
Eg① Class ~~Main~~ test
                    ②
  {
  Public static Void main(String[] args)
                      ③     ⑤     ④
   {
      int x = 10;
          ①
   }
  }
```

→ String is Class in Java.

→ totally 5 identifiers.

```
                 → Classname
Eg②:-  Class Test {
              ①          → method              variable name
   Public static Void main (String[] args){
                       ①   method    ③

        System.out.println("Sachin");
         class   ↓    method
               Variable
        (instance of System class)
```

→ totally 7 identifiers.

Eg ③

```
                                    → class name
Class Demo {
      ①
   Public static Void main (String[] args){
                      ②        ③       ④

        String name = 'sachin';
          ⑥     ⑤
        String result = name.toupperCase();
          ⑦     ⑧             ⑨
        System.out.Println(result);
          ⑩     ⑪    ⑫
   }
}
```

② .1 Rules for writing an indentifier
↳ w.r.t to (jvm + Compiler).

(i) Rule 1 :- The only allowed characters in java identifiers are a to z, A to Z, 0 to 9, _ (underscore), $.

② Rule 2 :- if we use any othe Character s it would result ~~[illegible]~~ $ as invalid (or in Compile time Error.

```
int ^* = 10; ~~[illegible]~~ (Invalid) X
int total = 10; (Valid) ✓
int total# = 10; (invalid) X
int telusko1 = 100; (Valid) ✓
```

~~[illegible]~~

Rule ③ :- identifiers are not allowed to start with digits.

```
Eg ① int telusko1 = 100; (valid) ✓
     int 1telusko = 100; (invalid) X
```

Rule ④ :- Java identifiers are Case sensitive.

```
Eg:- Class Demo {
         int number = 10;
         int Number = 20;
         int nUmber = 30;
         int NUMBER = 40;
     }
```

} all these Variable are different.

Rule ⑤ :- there is no length limit on java identifiers but still it is a good practice to keep the length of the identifier, no more than 15 character.

Eg ① int Priority of Thread With Min value = 1;

→ identifiers Can also start with Special Symbol '$' and '_'.

```
Eg ① int $a = 10;
   ② int _a = 10;
```

→ Eg : if, while, for, else, string, try Catch (or) throw averts.

(2.2) Reserved words

→ these are identifier which ~~are~~ have Special Meaning associated with. Compiler and JVM. (or it is a built in words/keywords which has already a predefined meaning to it.

→ they are Reserved for some Meaning.

Eg:- if, while, for, Else, String, try, Catch, throw, throws, assert, true, false, True, False.

→ they are Predefined for Compiler and JVM.

Rule ⑥:- We Can't use reserved Words as a identifiers

Eg:- int if =10;
↳ Compile time Error

Rule ⑦:- Predefined Class names Can be used as identifiers.

① String → inbuilt class name

② Runnable → interface (inbuilt) ~~Inbuilt~~

③ student → user defined Class name.

Eg① int string =10;

System.out.println(10);

Eg②- String Runnable = "Sachin";

System.out.println(Runnable);

Note:- Eventhough predefined Class names Can be used as an identifiers, it is not a good Practise to Keep.

→ if and IF different.
↳ Reserved word.

→ int int =10; X
S.o.P (int);
↳ Reserved word.
↳ Compile time Error.

$$\begin{tikzcd}
 & \boxed{\text{Reserved words(53)}} \arrow[dl] \arrow[dr] & \\
\text{Keyword(50)} \arrow[d] \arrow[dr] & & \text{Reserved literals(3)} \\
\text{used keywords(48)} & \text{unused (2) keyword} &
\end{tikzcd}$$

unused (2) keyword
↳ goto
↳ const

Reserved literals(3)
→ true → Values for boolean data types
→ false →
→ null → default Value for object reference

T:10 "38.

Literal :-

→ Any Constant value. which can be assigned to variable is Called literal

Eg ①:- int data = 10 //

$$\begin{tikzcd}
\text{int} \arrow[d] & \text{data} \arrow[d] & \text{10} \arrow[d] \\
\text{datatype / Reserved words.} & \text{Variable / identifier} & \text{literal}
\end{tikzcd}$$

→ Any Constant Value. which can be assigned to Variable is Called Literal.

Reserved literals (3)

① true
② false]→ Value of boolean datatypes
③ null → default Value for object reference

Eg ① boolean result = true;
↳ then literal is associated with boolean Variable

② boolean flag = false;

③ boolean ExamResult = True;
④ boolean Exam Result = true;]→ these different for JVM
⑤ boolean flag = 0; ↳ boolean literals

→ In C & C++ 0 → false, 1 → true

→ In Java '0 → means zero
1 → means zero

Note :- for boolean data types the only Values allowed for a Variable is "true" & "false", other than this if we try to keep any Values it would result in "Compile Time Error".