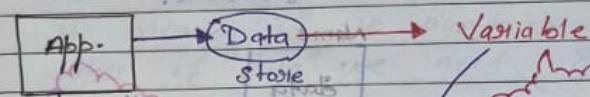


Topic 2 - Variable & Data Type

PAGE: _____
DATE: _____

"In the previous classes I had given you a scenario of Facebook let's recall it or take any other application whatever you want example of Instagram etc. Let us take your few social media Applications that run in any Application"



Ex: Weather Forecast
Google Map.

Mobile phone apps

Smartphones

Tablets

Laptops

Computers

Smartwatches

Smart TVs

Smart speakers

Smart home devices

Smart phones

Smart watches

Smart speakers

Smart home

Smart TV

Smart speakers

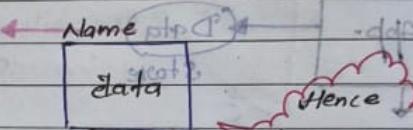
✓

Definition:- A variable is a named block of memory.

Advantage :-

→ we can store/fetch the data using name.

→ we can store only one value inside the Variable.



Hence

It is a small block of memory.

That's why either we can store Distance or speed Data if you want to store both,

You have to create one another Variable.

→ whatever the data we are going to store inside a Variable is not permanent so it is temporary.

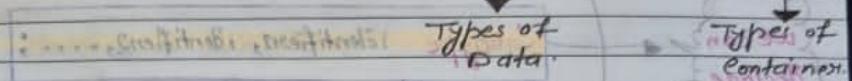
Every Variable having a "Lifespan" after certain time a Variable has to be destroyed (when it will destroy we will discuss further).

Every Variable has scope (Visibility).

* Data Type (which type of data)

" Let us understand with an example ok."

" Suppose I want to store
 " water we need Bottles.
 " copy " " Rag
 " clothes " " Almonds
 " Human " " House.

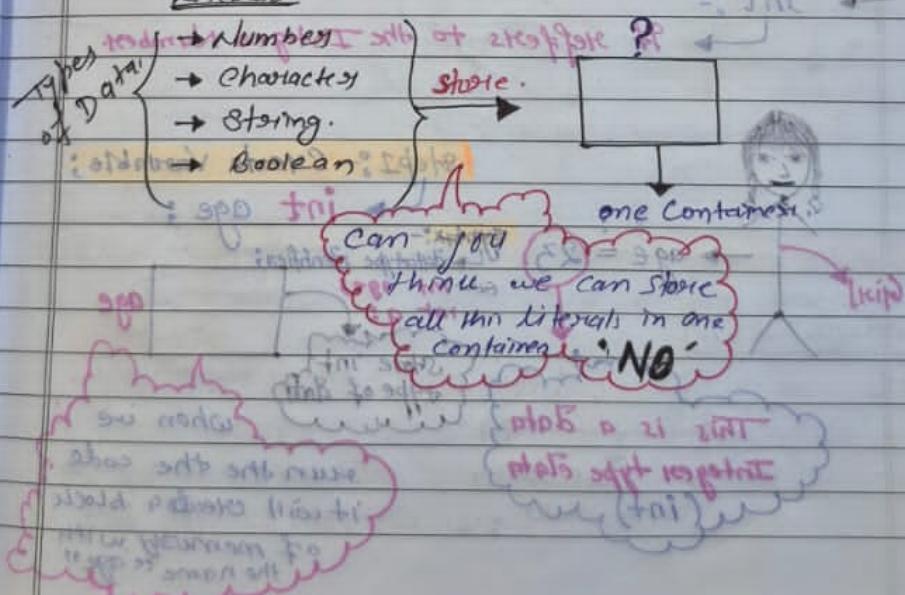


It means if we want to store different types of "data" we need diff diff types of containers"

(Let us correlate with our topic)

" We know that we have different different types of literals. What do you think we can store all this type of variables in the same container."

Literals



So, The syntax to declare a Variable is:-

Syntax :-

datatype identifier;

datatype identifier1, identifier2, ... ;

We can declare multiple variables in single line

(Variable Declaration Statement)

Example :- (Create Variable)

→ int :-

It refers to the Integer Number.



Gist

This is a Data
Integer type data
(int) age

Step1 :- Create Variable;

syntax:-
datatype identifier;
Ex:- int age;

Store int
type of data

age

when we
run the code
it will create a block
of memory with
the name "age".

Example :- (Store the Value in Variable)

Assignment operator

with the help
of assignment
operator we add the

Step 2 :- Add the Value to the container

age = 23 ;

Syntax :-

↳ identifier = Data ; Assignment Statement

Ex:- age = 23 ;

 age

23

Note :- Assignment operator work from "Right" to "Left" means whatever the value present inside the "Right side" will be stored to the "Left side Variable".

Ex:- age = 23 ;

Variable

= 23 ;

Storing

@ data/Value

Name / Array Name /

Object

Step 3 :- Declare & Initialization Statement in a Single Line.

Syntax :-

↳ datatype identifier = ^{Literals / Data} int age = 23 ;

.... ;

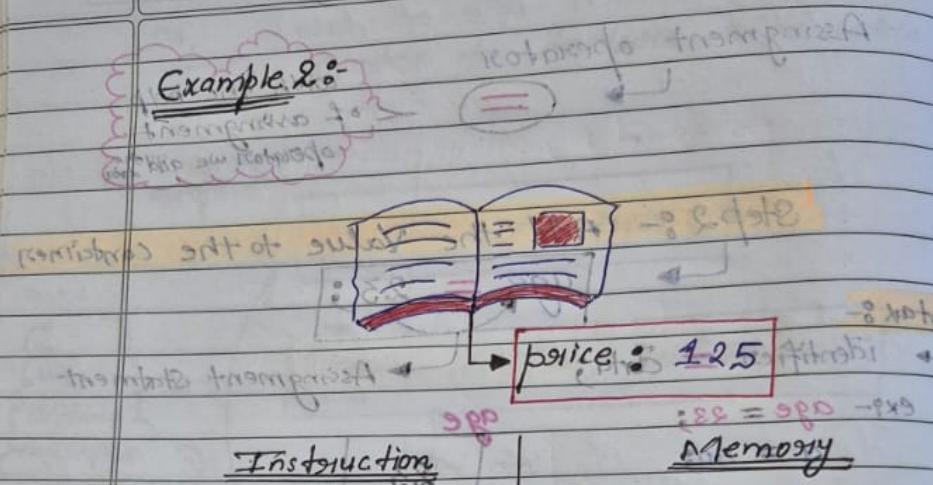
①

Ex:- int a=1, b=2 ;

Declaration

char ch1='a', ch2, ch3='b' ; Statement

(Adapted from Dr. S. R. Acharya)

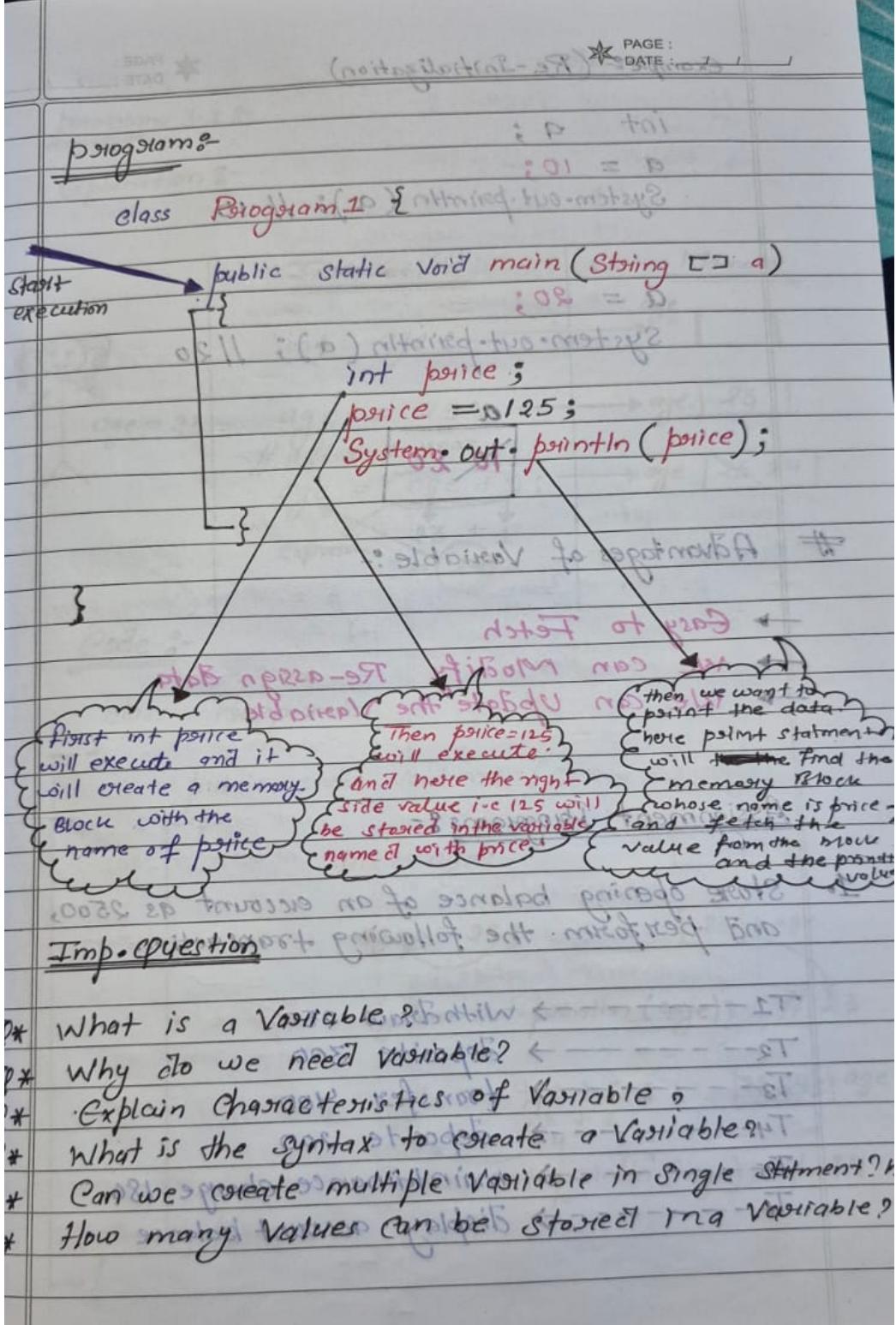
Example 2.0

→ int price; ✓ 1st
 or "tde" word shows memory transparency
 → Now price = 125; ✓ 2nd
 identifier → price, operator → =, value/data → 125
 operator → =, value/data → 125
 Literal/Value/data → 125

→ pointing
 System.out.println(price);
 Inside point Statement we pass the variable name, here point Statement will be the the memory block whose name is price and whatever the value present inside the price Variable in it will come to print statement and then print the value.

int i = 5; ✓ 1st
 Identifier → i, Operator → =, Value/Data → 5

int i = 5; ✓ 2nd
 Identifier → i, Operator → =, Value/Data → 5



Example:- (Re-Initialization)

```
int a;
```

a = 10;

System.out.println(a); // 10

-~~erroneous~~

a = 20;

System.out.println(a); // 20

: 20 a = 20

; (20) altered ~~10 20~~

Advantages of Variable :-

- Easy to Fetch
- We can Modify Re-assign data
- We can Update the Variable

Assignment Programs :-

1. Store opening balance of an account as 2500, and perform the following transaction.

T1 → Withdraw 2500

T2 → Deposite 300

T3 → Transfer 400

T4 → Deposite 200

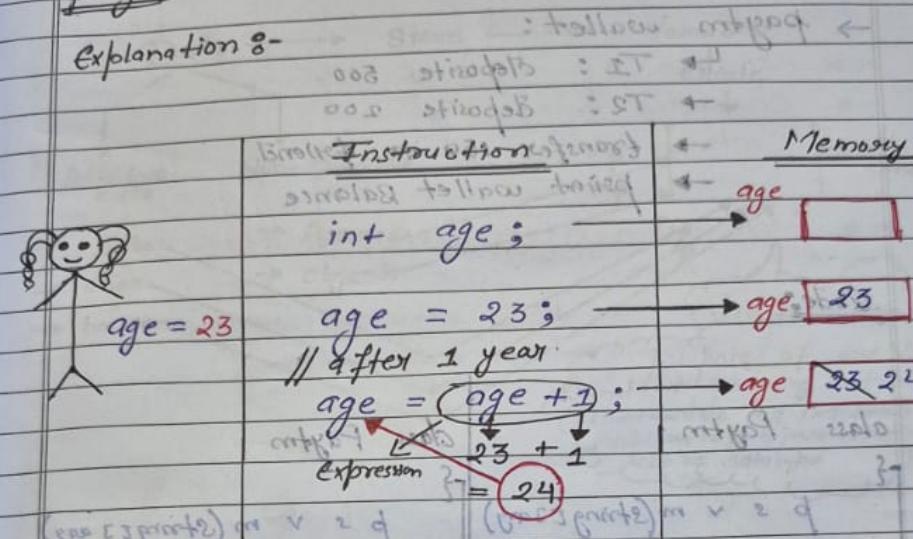
T5 → Maintenance charge 180

T6 → Display account balance

Program 3.18

- 8 pointers to memory

Explanation :-



Code :-

```

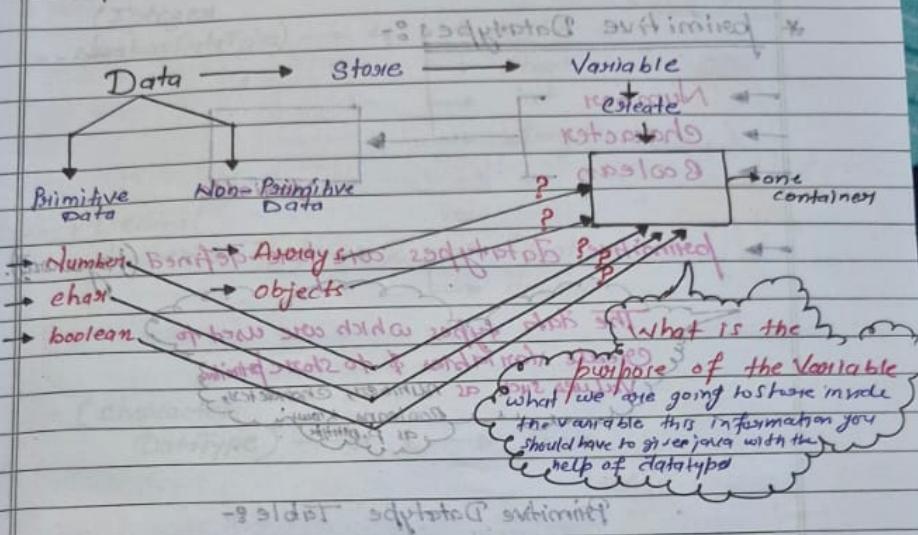
class Program2
{
    public static void main(String [] args)
    {
        int age;
        age = 23;
        age = age + 1;
        System.out.println(age); // 24
    }
}
  
```

The code is annotated with handwritten notes explaining the state of variables:

- Initial state: `age` is 23.
- After `age = age + 1;`: `age` is 24.
- Final output: `System.out.println(age);` prints 24.

Topic :- Data Types

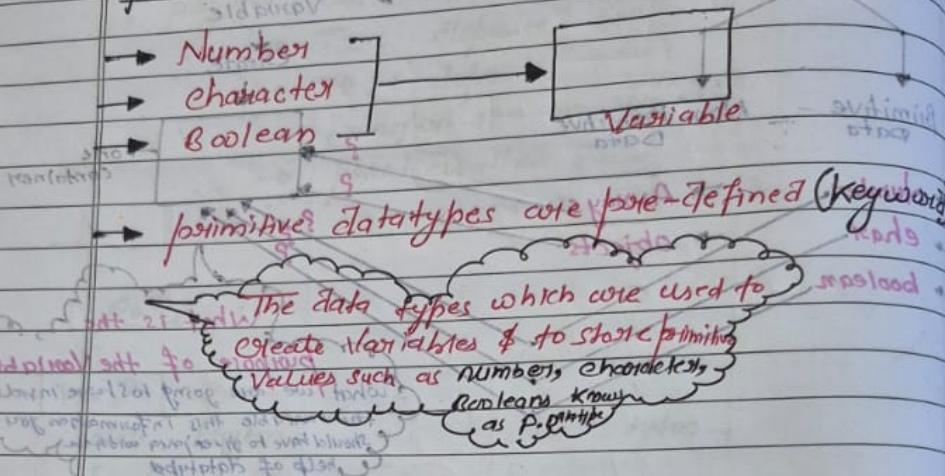
PAGE : _____
DATE : _____



Definition :-

| Size | datatype | store primitive | svimilie |
|----------|---|------------------|---------------|
| step 1 | 0 | step 1 | step 1 |
| step 2 | float | float primitive | float |
| step 3 | double | double primitive | double |
| step 4 | char | char primitive | char |
| step 5 | int | int primitive | int |
| step 6 | long | long primitive | long |
| step 7 | float | float | float |
| step 8 | double | double | double |
| step 9 | char | char | char |
| step 10 | int | int | int |
| step 11 | long | long | long |
| step 12 | primitive | non-primitive | non-primitive |
| step 13 | Datatype | Datatype | Datatype |
| step 14 | Enlargement of size of data read from INT | - store | - |
| step 15 | float | float | float |
| step 16 | double | double | double |
| step 17 | char | char | char |
| step 18 | int | int | int |
| step 19 | long | long | long |
| step 20 | float | float | float |
| step 21 | double | double | double |
| step 22 | char | char | char |
| step 23 | int | int | int |
| step 24 | long | long | long |
| step 25 | float | float | float |
| step 26 | double | double | double |
| step 27 | char | char | char |
| step 28 | int | int | int |
| step 29 | long | long | long |
| step 30 | float | float | float |
| step 31 | double | double | double |
| step 32 | char | char | char |
| step 33 | int | int | int |
| step 34 | long | long | long |
| step 35 | float | float | float |
| step 36 | double | double | double |
| step 37 | char | char | char |
| step 38 | int | int | int |
| step 39 | long | long | long |
| step 40 | float | float | float |
| step 41 | double | double | double |
| step 42 | char | char | char |
| step 43 | int | int | int |
| step 44 | long | long | long |
| step 45 | float | float | float |
| step 46 | double | double | double |
| step 47 | char | char | char |
| step 48 | int | int | int |
| step 49 | long | long | long |
| step 50 | float | float | float |
| step 51 | double | double | double |
| step 52 | char | char | char |
| step 53 | int | int | int |
| step 54 | long | long | long |
| step 55 | float | float | float |
| step 56 | double | double | double |
| step 57 | char | char | char |
| step 58 | int | int | int |
| step 59 | long | long | long |
| step 60 | float | float | float |
| step 61 | double | double | double |
| step 62 | char | char | char |
| step 63 | int | int | int |
| step 64 | long | long | long |
| step 65 | float | float | float |
| step 66 | double | double | double |
| step 67 | char | char | char |
| step 68 | int | int | int |
| step 69 | long | long | long |
| step 70 | float | float | float |
| step 71 | double | double | double |
| step 72 | char | char | char |
| step 73 | int | int | int |
| step 74 | long | long | long |
| step 75 | float | float | float |
| step 76 | double | double | double |
| step 77 | char | char | char |
| step 78 | int | int | int |
| step 79 | long | long | long |
| step 80 | float | float | float |
| step 81 | double | double | double |
| step 82 | char | char | char |
| step 83 | int | int | int |
| step 84 | long | long | long |
| step 85 | float | float | float |
| step 86 | double | double | double |
| step 87 | char | char | char |
| step 88 | int | int | int |
| step 89 | long | long | long |
| step 90 | float | float | float |
| step 91 | double | double | double |
| step 92 | char | char | char |
| step 93 | int | int | int |
| step 94 | long | long | long |
| step 95 | float | float | float |
| step 96 | double | double | double |
| step 97 | char | char | char |
| step 98 | int | int | int |
| step 99 | long | long | long |
| step 100 | float | float | float |
| step 101 | double | double | double |
| step 102 | char | char | char |
| step 103 | int | int | int |
| step 104 | long | long | long |
| step 105 | float | float | float |
| step 106 | double | double | double |
| step 107 | char | char | char |
| step 108 | int | int | int |
| step 109 | long | long | long |
| step 110 | float | float | float |
| step 111 | double | double | double |
| step 112 | char | char | char |
| step 113 | int | int | int |
| step 114 | long | long | long |
| step 115 | float | float | float |
| step 116 | double | double | double |
| step 117 | char | char | char |
| step 118 | int | int | int |
| step 119 | long | long | long |
| step 120 | float | float | float |
| step 121 | double | double | double |
| step 122 | char | char | char |
| step 123 | int | int | int |
| step 124 | long | long | long |
| step 125 | float | float | float |
| step 126 | double | double | double |
| step 127 | char | char | char |
| step 128 | int | int | int |
| step 129 | long | long | long |
| step 130 | float | float | float |
| step 131 | double | double | double |
| step 132 | char | char | char |
| step 133 | int | int | int |
| step 134 | long | long | long |
| step 135 | float | float | float |
| step 136 | double | double | double |
| step 137 | char | char | char |
| step 138 | int | int | int |
| step 139 | long | long | long |
| step 140 | float | float | float |
| step 141 | double | double | double |
| step 142 | char | char | char |
| step 143 | int | int | int |
| step 144 | long | long | long |
| step 145 | float | float | float |
| step 146 | double | double | double |
| step 147 | char | char | char |
| step 148 | int | int | int |
| step 149 | long | long | long |
| step 150 | float | float | float |
| step 151 | double | double | double |
| step 152 | char | char | char |
| step 153 | int | int | int |
| step 154 | long | long | long |
| step 155 | float | float | float |
| step 156 | double | double | double |
| step 157 | char | char | char |
| step 158 | int | int | int |
| step 159 | long | long | long |
| step 160 | float | float | float |
| step 161 | double | double | double |
| step 162 | char | char | char |
| step 163 | int | int | int |
| step 164 | long | long | long |
| step 165 | float | float | float |
| step 166 | double | double | double |
| step 167 | char | char | char |
| step 168 | int | int | int |
| step 169 | long | long | long |
| step 170 | float | float | float |
| step 171 | double | double | double |
| step 172 | char | char | char |
| step 173 | int | int | int |
| step 174 | long | long | long |
| step 175 | float | float | float |
| step 176 | double | double | double |
| step 177 | char | char | char |
| step 178 | int | int | int |
| step 179 | long | long | long |
| step 180 | float | float | float |
| step 181 | double | double | double |
| step 182 | char | char | char |
| step 183 | int | int | int |
| step 184 | long | long | long |
| step 185 | float | float | float |
| step 186 | double | double | double |
| step 187 | char | char | char |
| step 188 | int | int | int |
| step 189 | long | long | long |
| step 190 | float | float | float |
| step 191 | double | double | double |
| step 192 | char | char | char |
| step 193 | int | int | int |
| step 194 | long | long | long |
| step 195 | float | float | float |
| step 196 | double | double | double |
| step 197 | char | char | char |
| step 198 | int | int | int |
| step 199 | long | long | long |
| step 200 | float | float | float |
| step 201 | double | double | double |
| step 202 | char | char | char |
| step 203 | int | int | int |
| step 204 | long | long | long |
| step 205 | float | float | float |
| step 206 | double | double | double |
| step 207 | char | char | char |
| step 208 | int | int | int |
| step 209 | long | long | long |
| step 210 | float | float | float |
| step 211 | double | double | double |
| step 212 | char | char | char |
| step 213 | int | int | int |
| step 214 | long | long | long |
| step 215 | float | float | float |
| step 216 | double | double | double |
| step 217 | char | char | char |
| step 218 | int | int | int |
| step 219 | long | long | long |
| step 220 | float | float | float |
| step 221 | double | double | double |
| step 222 | char | char | char |
| step 223 | int | int | int |
| step 224 | long | long | long |
| step 225 | float | float | float |
| step 226 | double | double | double |
| step 227 | char | char | char |
| step 228 | int | int | int |
| step 229 | long | long | long |
| step 230 | float | float | float |
| step 231 | double | double | double |
| step 232 | char | char | char |
| step 233 | int | int | int |
| step 234 | long | long | long |
| step 235 | float | float | float |
| step 236 | double | double | double |
| step 237 | char | char | char |
| step 238 | int | int | int |
| step 239 | long | long | long |
| step 240 | float | float | float |
| step 241 | double | double | double |
| step 242 | char | char | char |
| step 243 | int | int | int |
| step 244 | long | long | long |
| step 245 | float | float | float |
| step 246 | double | double | double |
| step 247 | char | char | char |
| step 248 | int | int | int |
| step 249 | long | long | long |
| step 250 | float | float | float |
| step 251 | double | double | double |
| step 252 | char | char | char |
| step 253 | int | int | int |
| step 254 | long | long | long |
| step 255 | float | float | float |
| step 256 | double | double | double |
| step 257 | char | char | char |
| step 258 | int | int | int |
| step 259 | long | long | long |
| step 260 | float | float | float |
| step 261 | double | double | double |
| step 262 | char | char | char |
| step 263 | int | int | int |
| step 264 | long | long | long |
| step 265 | float | float | float |
| step 266 | double | double | double |
| step 267 | char | char | char |
| step 268 | int | int | int |
| step 269 | long | long | long |
| step 270 | float | float | float |
| step 271 | double | double | double |
| step 272 | char | char | char |
| step 273 | int | int | int |
| step 274 | long | long | long |
| step 275 | float | float | float |
| step 276 | double | double | double |
| step 277 | char | char | char |
| step 278 | int | int | int |
| step 279 | long | long | long |
| step 280 | float | float | float |
| step 281 | double | double | double |
| step 282 | char | char | char |
| step 283 | int | int | int |
| step 284 | long | long | long |
| step 285 | float | float | float |
| step 286 | double | double | double |
| step 287 | char | char | char |
| step 288 | int | int | int |
| step 289 | long | long | long |
| step 290 | float | float | float |
| step 291 | double | double | double |
| step 292 | char | char | char |
| step 293 | int | int | int |
| step 294 | long | long | long |
| step 295 | float | float | float |
| step 296 | double | double | double |
| step 297 | char | char | char |
| step 298 | int | int | int |
| step 299 | long | long | long |
| step 300 | float | float | float |
| step 301 | double | double | double |
| step 302 | char | char | char |
| step 303 | int | int | int |
| step 304 | long | long | long |
| step 305 | float | float | float |
| step 306 | double | double | double |
| step 307 | char | char | char |
| step 308 | int | int | int |
| step 309 | long | long | long |
| step 310 | float | float | float |
| step 311 | double | double | double |
| step 312 | char | char | char |
| step 313 | int | int | int |
| step 314 | long | long | long |
| step 315 | float | float | float |
| step 316 | double | double | double |
| step 317 | char | char | char |
| step 318 | int | int | int |
| step 319 | long | long | long |
| step 320 | float | float | float |
| step 321 | double | double | double |
| step 322 | char | char | char |
| step 323 | int | int | int |
| step 324 | long | long | long |
| step 325 | float | float | float |
| step 326 | double | double | double |
| step 327 | char | char | char |
| step 328 | int | int | int |
| step 329 | long | long | long |
| step 330 | float | float | float |
| step 331 | double | double | double |

* Primitive Datatypes :-



Primitive Datatype Table :-

| Primitive Value | Primitive Data Types | Default Values | Size |
|-----------------|---|------------------------------|--------------------------------------|
| Numbers | Integer (Whole Numbers) +ve to 0 to -ve | byte short int long | 1 byte 2 byte 4 byte 8 byte |
| Floating Values | float double | 0.0f/F 0.0D/D | 4 byte 8 byte |
| Character | char | '\u0000' | 2 byte |
| Boolean | boolean | false | 1 bit |

Note:- The number data type in increasing order of the capacity.

byte < short < int < long < float < double

(Integer)
NumberDataType) → 4
i.e. int, short, long, byte

- byte
- short
- int
- long



(Decimal)

NumberDataType) → 2

- float
- double

i.e. float, double

(Character)

DataType) → 1 → char

(Boolean) i.e. true, false, boolean

DataType) → 1 → boolean

8 → total no. of primitive
datatype.

Since primitive DataTypes are pre-defined or it is a keyword so,

Every primitive datatype must be written in lower case.

Ex:-

i(2015) attited - two mistakes

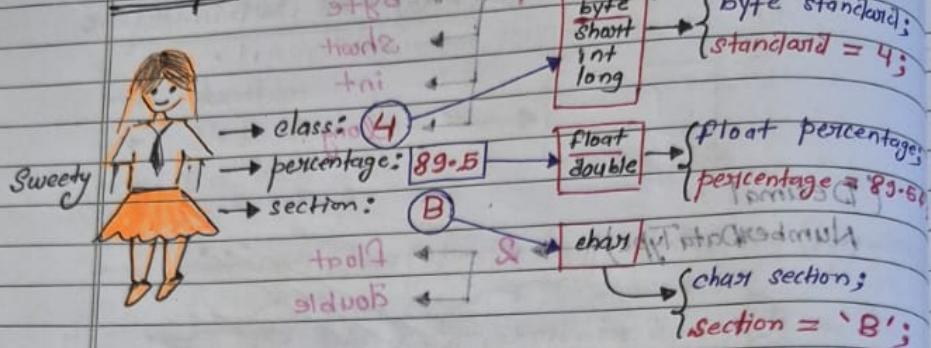
i(12) Shorrtened - two mistakes ✓

i(12) attited - two mistakes

Int X

int ✓

Example :-



Code :-

```
records ← ↓ ← (adaptTo)
class School
{
    public static void main (String args[])
    {
        byte clas; 8
        clas = 4;

        float per;
        per = 89.5f;
        System.out.println (clas);
        System.out.println (per);
        System.out.println (sec);
    }
}
```

System.out.println (clas);

System.out.println (per);

System.out.println (sec);

tni

X tni

1

Difference b/w each Number DataTypes :-
capacity :-

| | | |
|-------|----------------------------------|---------------------------------|
| byte | use to store Integer value | (255) ^{1 byte} |
| short | | (65535) ^{2 bytes} |
| int | | (2147483647) ^{4 bytes} |

long long long

float float

double double

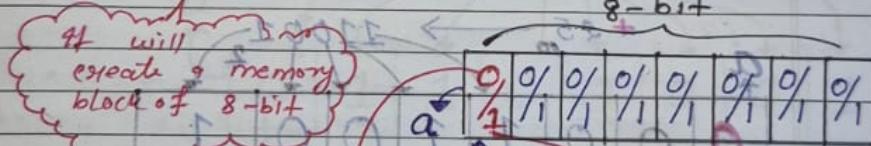
How Data will store inside the memory Let us understand by an example :-

2 Data type must identify Size of datatype

byte a ;

1 byte.

8-bit



If the we store
positive no.
then
first block it will
store '0'

Sign Bit

If the no.
is a negative.
Number then
it will store
'1'

If it is the
representation
of the no. that
is true or -ve

Binary representation starts from
left most part point zero
starts on sign bit

→ Suppose if i want to store 25 in byte
→ data type

+ve (25)
 $\frac{25}{10}$

Base is 10 because it is decimal no.

(010111)
start from right to left
we need to convert it into binary No.

→ How to convert Decimal to binary?

+ve no so sign bit is '0'

2 | 25
2 | 12
2 | 6
2 | 3
1 | 1

Binary Formate of 25

25 → 11001

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|

Because 25 is +ve number

Rest of the block filled with 0

This is how the data will get stored inside the memory and same thing happen for all the no. data type

Capacity (Range)

1) byte:

→ min: -128

→ max: 127

2) short:

→ min: -32768

→ max: 32767

3) int:

→ min: -2147483648

→ max: 2147483647

4) long:

→ min: -9223372036854775808

→ max: 9223372036854775807

5) float:

→ min: 1.4E-45

→ max: 3.4028235638

6) double:

→ min: 4.9 - 324

→ max: 1.7976931348623157E308

* Non-Primitive Data type :-

11) **purpose**
The main purpose of Non-primitive variable is to store reference of address of another block of memory.

Non-primitive data is multi-valued data. To store this, we need more than 1 memory block.

① Class **Box** → ① "Box is the name of class"
{} ↗ ② "It is also a Non-primitive
 datatype" ↗

Note:- In Java we don't have anything which is pre-defin.
Non-primitive data-types

- establish standards of ~~mod~~ every class we create
the same for all classes

N 10011 non-primitive (datatype)

11) Glass Book

1). Book is the name of class

2). Book is also a Non-primitive data type.

Every class Name in java is non primitive data type + 8 + 0 + 0 + 1

~~Ex^o- class~~

$$\rightarrow \text{H}_2\text{SO}_4 \text{ is neutral}$$

→ Q, Sir Can i create non-primitive variable by using Non-primitive datatype? "Yes we can"

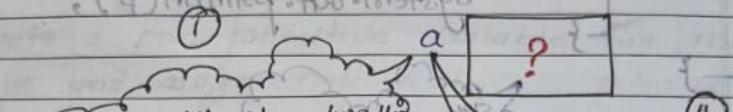
class Book

Syntax:-

```
{          datatype identifier;
    }        Book a;
```

}

: (d) nited two - mable



"a" will store two things.

- 1. null
- 2. reference of the

Book object

Hence it is also stored

reference of the Book object

so this variable is

known as the

reference variable

Code:-

class Book {

}

class Program1

{ to not show tht ob jett is known as "null"

public static void main(Staring args)

{

Book b1, b2, b3;

b1 = b2 = b3 = null;

? do it print System.out.println(b); // null

System.out.println(b1); // null

? print al ob jett System.out.println(b2); // null

System.out.println(b3); // null

? print all ob jett System.out.println(b); // null

System.out.println(b1); // null

System.out.println(b2); // null

System.out.println(b3); // null

class Program {
 public static void main(String args) {
 System.out.println("Hello World");
 }
}

can i create the
the variable of scrum
class who consist the main
method?

Conclusion:-

- Every class Name is a Non-primitive data type.
 - "null" keyword is the default value for all (non-primitive) variable data types.
 - We can create reference Variable with the help of non-primitive data types.

Imp. questions - pt. II

- * What is non-primitive datatype?
 - * Why do we need non-primitive datatype?
 - * What is null?
 - * How to create reference variable in java?
 - * What is a class name in java?

Assignment Programs :-

1) Create a non-primitive variable for bottle type and assign null in it.

2) Create a non-primitive variable for pen and assign null in it.

3) Create a non-primitive variable for movie type and assign null in it.

* Difference between primitive datatype and Non-primitive datatype.

| primitive Datatype | Non-primitive Datatype |
|---------------------------------|---|
| → to create primitive variables | → to create Non-primitive or reference variable |

→ primitive Datatype besides primitive non-primitive

→ User-defined is not allowed.
(pre-defined)

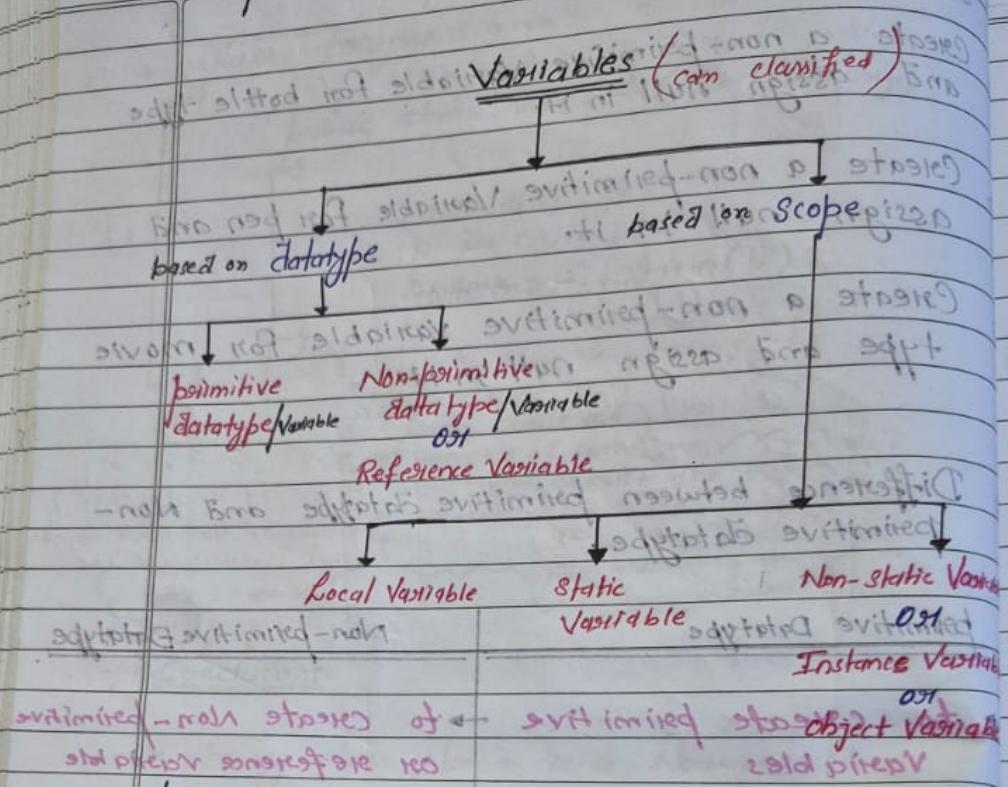
User-defined is allowed.

int sum() { int a = 10; int b = 20; return a + b; }

int sum() { int a = 10; int b = 20; return a + b; }

int sum() { int a = 10; int b = 20; return a + b; }

Topic 8 - Types of Variable



* Primitive Variable :-

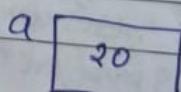
• primitive variable → created by primitive +
• primitive variable → primitive variable

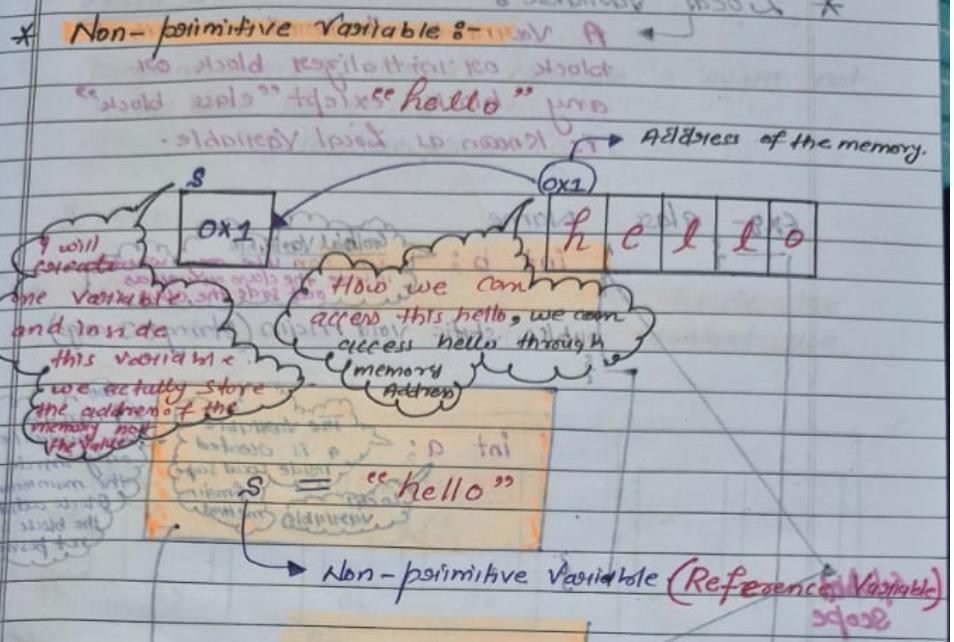
- primitive variable → primitive Data-type

→ Values are stored inside the primitive variable.

Ex:- int (a); → primitive variable.

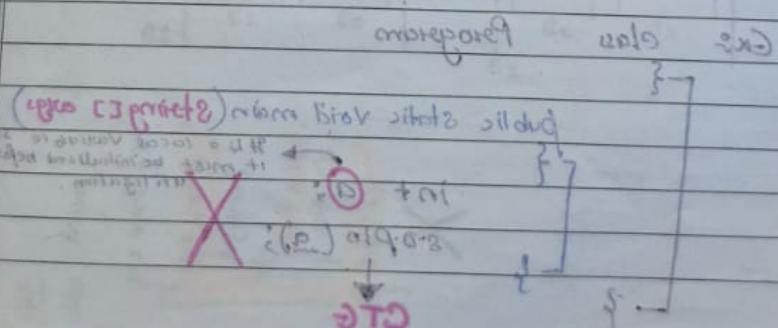
$$a = 20;$$





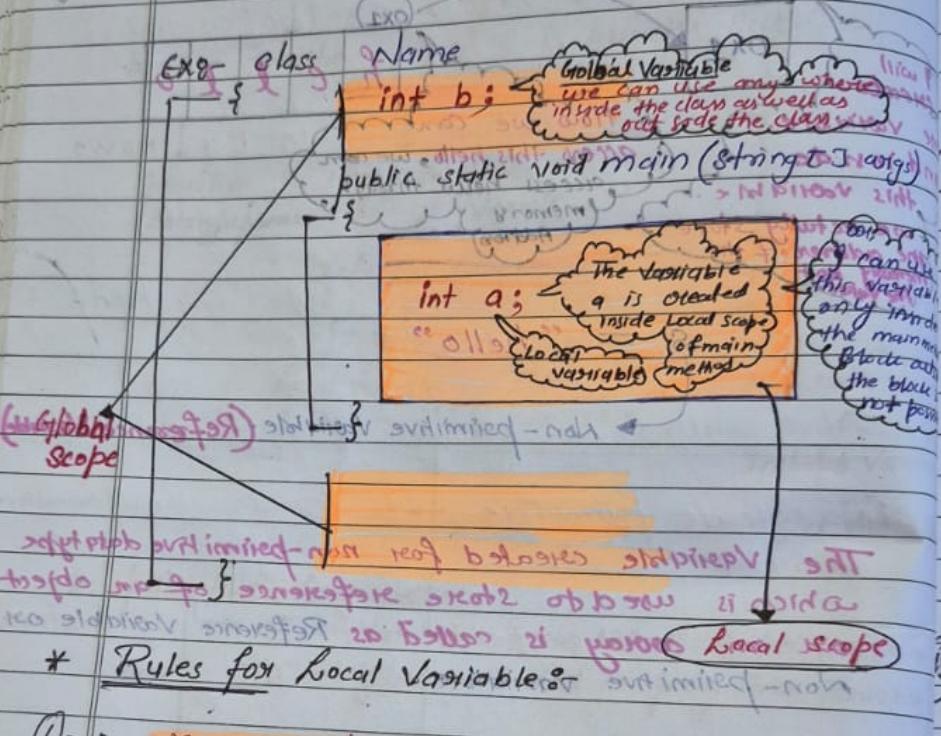
The variable created for non-primitive datatype which is used to store reference of an object (array and array) is called as Reference Variable or Non-primitive Variable.

Variables *s1*, *s2*, *s3* are for arrays \rightarrow (1)
arrays are primitive



* Local Variable :

→ A variable declared in a method block or initializer block or any "block" except "class block" is known as Local Variable.



* Rules for Local Variable :

- ① → we can not use local variable without assigning data

Exo class Program

public static void main (String [] args)

int a;
it is a local variable so it must be initialized before utilization.

s.o.p (a);
CTG

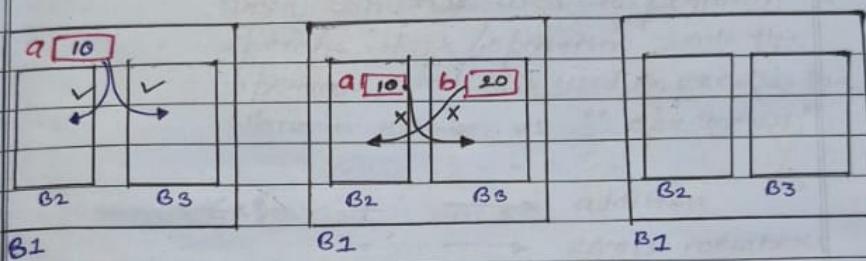
e9191031 :-

not scope. do it

program.java : 5: error: variable a might not have been initialized. value set by prior assignment to System.out.println(a);

1 e9191031

- (ii) → we can use the variable only inside the block where it is declared (we cannot use outside the block)



- (iii) → we can not declare 2 local variable with same name inside the same scope.

