Lab

Q. class A
{
public static void main(String[] args)
{
System.out.printf("angshul");
}
}

String... ar
↑or

A. in c, 1 = true, 0 = false
in Java boolean value is there. So in 'if' condition boolean value required.
Int will give error.

8. odd even check.

```
import java.util.Scanner;
class odev{
public static void main (String[] args){
Scanner (sc) = new Scanner (System.in);
System.out.printf ("Enter num: ");
int n = (sc).nextInt();
if (n%2==0){
        system.out.printf ("even");
}
else{ System.out.printf ("odd"); }
}
}
```

14/1/25

Variable

types:

1) Local variable
2) Instance variable
3) Static variable.

data type

byte → 1byte/8bit
short → 2byte
int → 4byte
long → 8byte
(Integral type of dt)

float → 4byte
double → 8byte
(floating type of dt)

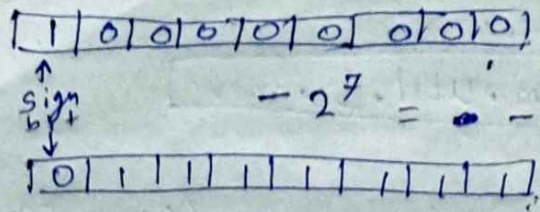
char → 2byte
boolean → 1bit

```
class A {
    static int a = 20;      → static variable
    int a = 20;             → Instance variable

    p s v m (string ..a){

    int a = 20;             → local variable
    }
```

8 bit range



$-2^7 = -128$

$2^7 - 1 = 127$

byte (-128 to 127)

Q.
```
{
    byte b1 = 125;
    byte b2 = 5;
    byte c = b1 + b2;   error:
    sop(c);
}
```

int operation compiler will op integer value.

c = byte(b1+b2)

Q.
```
byte b1 = 120;
byte b2 = 30;
byte c = byte(b1+b2)
       = ?
```

120+30 = 150

127 + 1 ⇒ -128

127 + 2 ⇒ -128 + 1

127 + 3 ⇒ -128 + 2

127 + 23 ⇒ -128 + 22 ⇒ -106

-128, -127, -126 ....

0, 1, 2, 3 ..... 126, 127 | 150

(*) Java support unicode character

C " ASCII "

(*) Double store approx value, not exact value.

float f = 5.3

5.3 → double, f → convert into float.

```
float f = 5.3;
if (f == 5.3)
    sop(Hi);  X
else
    sop(Hello);  X
```

16/1/25

```
class A
{
    static int a= 50;
    public static void main(String[] args)
    {
        int a=40;
        method1();
        System.out.println(a);
    }
    static void method1()
    {
        system.out.println("M1");
    }
}
```

syntax of creating object

```
A a1 = new A()
Sys.out.pr(a1.a)
```

→ Constructor

Types of constructor:

① Default Constructor.

```
A()  → 0 argument constructor.
{
}
```

② 0 argument Constructor.

```
A()
{
}
```

③ Parameterized Constructor.

```
A(int a, int b){
...
}
```