JAVA

SMALLEST UNITS OF
ENGLISH LANGAUGE

Tokens:

The smallest component/ samllest units of any programming language

1)keywords
2)identifiers
3)literals / values

keywords:

pre-defined words

run

dance

sleep

→ Understandable by the Compiler

→ Reserved words

↓

we cannot change the meaning of that words

Rules:

1)keywords should be written in Lower Case

class ✓ Class ✗

public ✓ Public ✗

2)identifiers:

variables
methods
class
interface
enum

The names given to the components of java like classes,variables,methods, interfaces,enum.....etc

ex:
```
class Program        identifier        identifier
{
public static void main(String[] args)
{
                                        identifier
}                    identifier

}
```

Rules :

1)identifiers should not start with Number

EX: Chaitanya1 ✓

1Dinga ✗

Dingi5 ✓

567Romeo ✗

Juliet5 ✓

2)Only $ and _ are allowed

Ex: $Bahubali ✓

Pushpa2@A ✗

Srivalli_5 ✓

Shikavath#2 ✗

5Amaron$ ✗

3)we cannot use keywords as Identifiers

Ex: class ✗

static ✗

CTE	Compile Time Successful

class class
{
 	
}
	class Class
	{
	upper case
	
	}

Conventions: → Industry Standards

Note : Compile does not check for Conventions

1) Conventions for ClassName

Ex: dinga ✗ Dinga ✓

dingi ✗ Dingi ✓

Dingadingi ✗ DingaDingi ✓

romeojuliet ✗ RomeoJuliet ✓

2)Conventions for methods:

Ex: method ✓ Method ✗

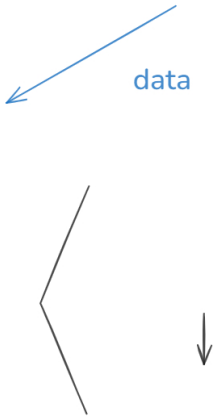
addition ✓ Addition ✗

addnumbers ✗ addNumbers ✓

addthreenumbers ✗ addThreeNumbers ✓

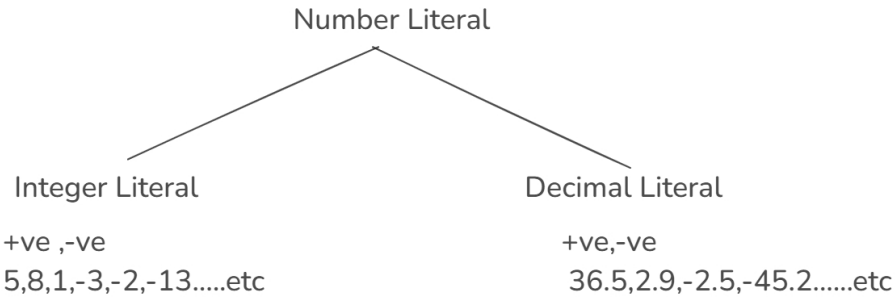
Literals:

Dinga



The data which is written by the programmer inside a program
Is known as Literal

There are 4 typesof literals
1)Number Literal
2)Character Literal
3)String Literal
4)Boolean Literal



Character Literal:

-->anything enclosed within SingleQuote ' '
is known as Character Literal
-->The length of Character Literal is always 1

Ex : a ✗

'a' ⟶ character literal

'd' ⟶ character literal

'ab' ✗

1 ⟶ integer literal

'1' ⟶ character literal

'12' ✗

'$' ⟶ character literal

String Literal:

-->anything enclosed within DoubleQuote " " is known as String Literal

-->The length of String literal can be anything

Ex : Dingi ✗

"Dingi" ⟶ String Literal

123 ⟶ Integer literal

"123" ⟶ String literal

"#Dinga5$" ⟶ String Literal

Boolean Literal:

Dinga

Are you committed with Dingi??

No

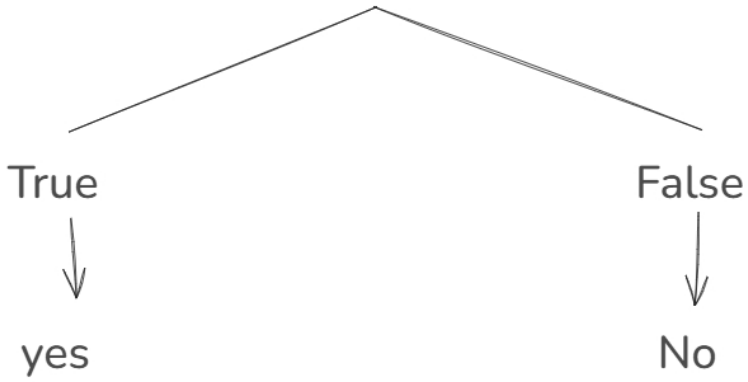
Are you guyz Active??

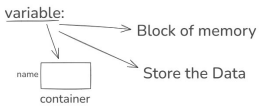
Yes

Had your Dinner??

No

To Represent Logical Data

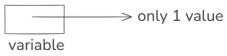




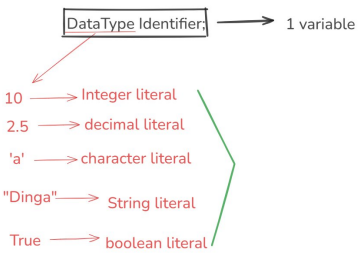
Variable is a named block of memory which is used to store the data

Advantages of Variable:

we can Store the Data or Fetch the data using name



Syntax of Variable:



Example to create a Variable:

age : 25

step 1 : Create a variable/ declare a variable

int age;

Step 2 : Initialize the variable

store 25

identifier=value;

age = 25;

Example :

price : 250

int price;//declaration
price=250;//initialise

Declaring and Intiliasing In single Line:

datatype Identifier=value;

int price=250;

Ex : int a;
a=10;

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

a=20;

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

-->we can update the variable
-->we can re-initialize the variable
or re-assign the data

Task :

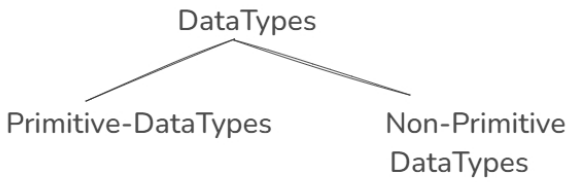
age : 27

Juliet

1)Print the present age
2)Print the age after one year
3)Print the age after 5 years

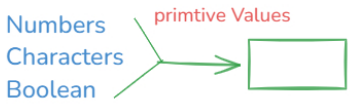
Paytm wallet
-->intilially 0 rupees
-->Deposit of 500
-->Deposit of 250
-->Transfer of 150
??Print the remaining money in the wallet

DataTypes: It Specifies what type of Data to be stored inside the variable



Primitive DataTypes:

-->The DataTypes which are used to create the Variables to store Primtive Values .



-->Primitive-DataTypes are Pre-Defined

keyword

-->They are used to store the primitive Values
-->There are 8 Primitive Datatypes

Primitive Values	Primtive DataTypes	Default Value	Size
Integer (+ve,-ve numbers)	byte	0	1byte
	short	0	2byte
	int	0	4byte
	long	0l/L	8byte
Floating Values	float	0.0f/F	4byte
	double	0.0d/D	8byte
Characters	char	/u0000	2byte
Boolean	boolean	False	1bit

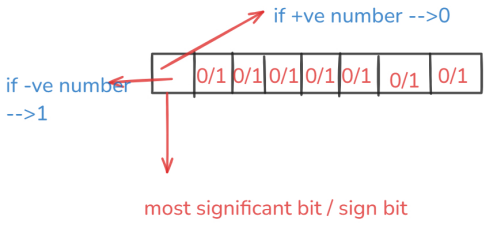
To Store the Integers - 4 DataTypes
To Store the Decimal Values - 2 DataTypes
To Store Characters - 1 DataType
To Store Boolean - 1 DataType

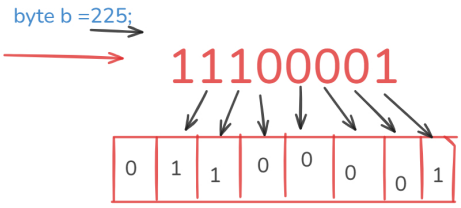
Task:

Rocky

clas : 9 → byte,short,int,long
percentage : 35.5 → float,double
Section : c → char

1 byte = 8 bits





Non-Primitive DataTypes :

```
class  Monday
  {


  }
```

Non-Primitive DataType

```
class Book
{


}
```

Non-primitive DataType

Can i create a variable for Non-Primitive Data Type:

```
class Monday
{


}
```

Syntax : DataType Identifier;

Monday a;

a ?

we can store

→ null(keyword)

→ reference / address of Monday Object of class type

```
class Instagram
{


}
```

DataType Identifier;

Instagram a ;

non-primitive variable

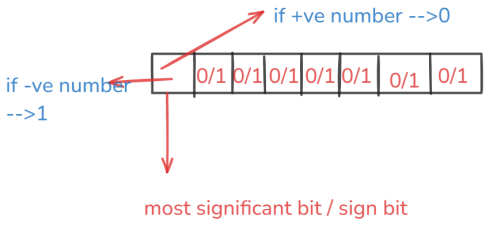
a ?

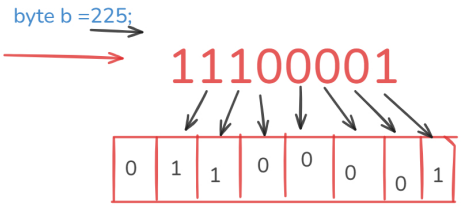
null

Reference / address

```
Ex : class Instagram
     {
      public static void main(String[] args)
      {
        Instagram a;
        a=10;
         a=null;

      }
     }
```

1 byte = 8 bits





Non-Primitive DataTypes :

```
class  Monday
  {


  }
```

Non-Primitive DataType

```
class Book
{


}
```

Non-primitive DataType

Can i create a variable for Non-Primitive Data Type:

```
class Monday
{


}
```

Syntax : DataType Identifier;

Monday a;

a ?

we can store

→ null(keyword)

→ reference / address of Monday Object of class type

```
class Instagram
{


}
```

DataType Identifier;

Instagram a ;

non-primitive variable

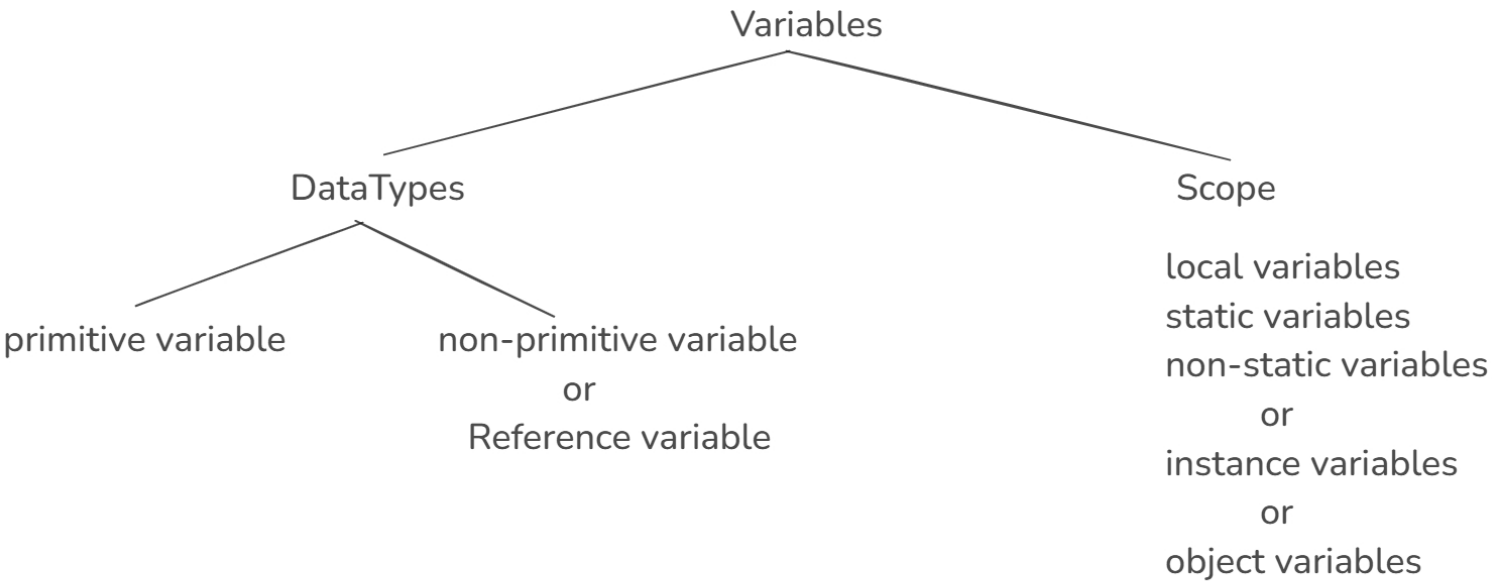
a ?

null

Reference / address

```
Ex : class Instagram
     {
      public static void main(String[] args)
      {
        Instagram a;
        a=10;
         a=null;

      }
     }
```

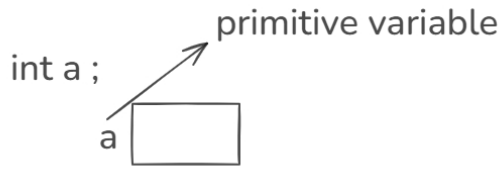
Types of Variables:



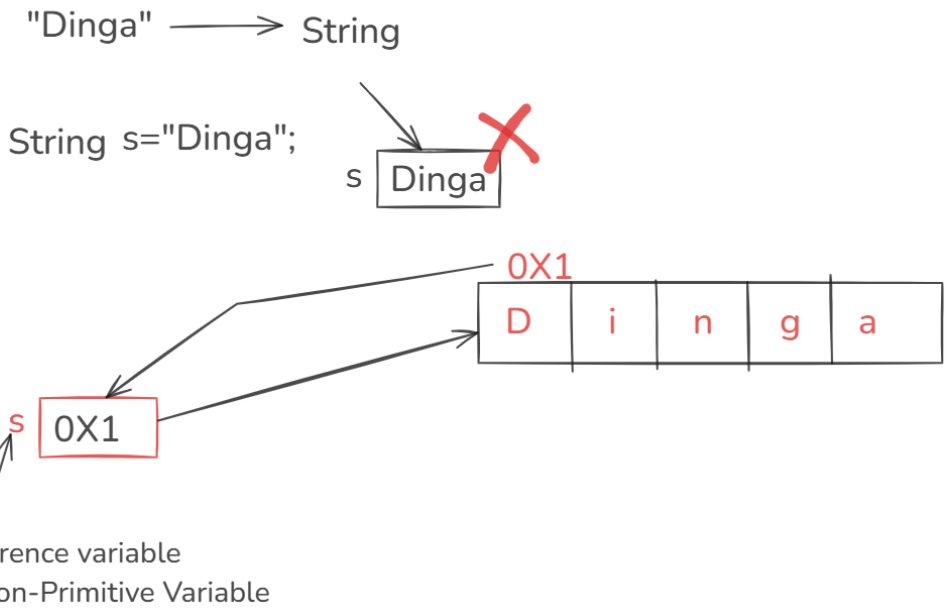
Primitive Variables :

The variables which are created for Primitive Datatypes are called as Primitive variables

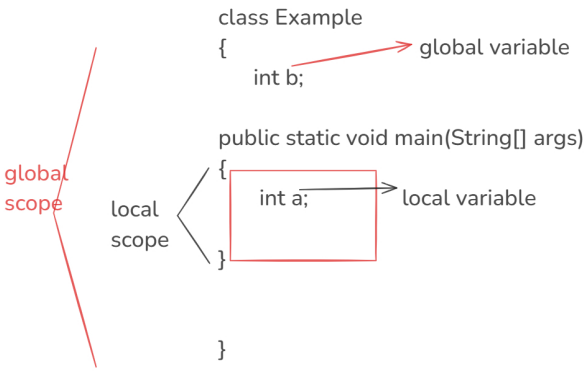
integer literals
decimal literals
character literals
boolean literals



Non-Primitive Variable:



Local Variable:



→ A variable which is declared inside the method block or any other block except the class block is known as Local Variable

Rules of Local Variable:

1)we cannot use the local variables without assigning the data

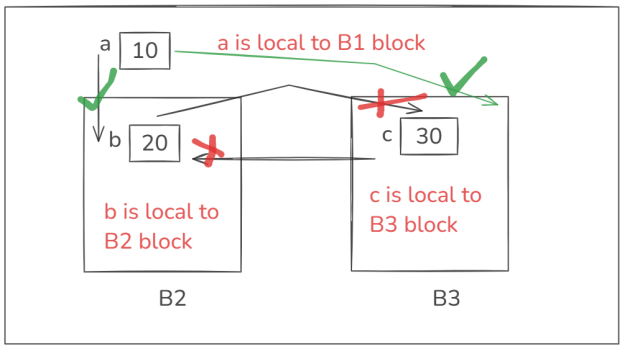
→ we cannot use them directly
→ no default value for local variables

Ex : class Hello
{
public static void main(String[] args)
{
int a; a
System.out.println(a); CTE

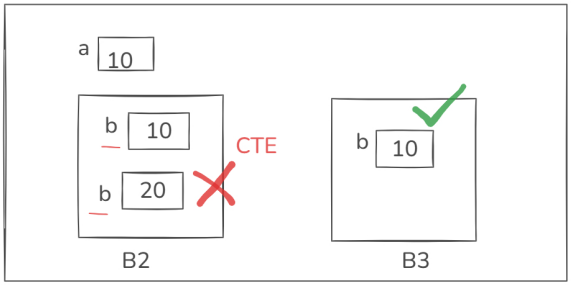
}

}

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



3)we cannot declare two local variables with same name inside the same scope

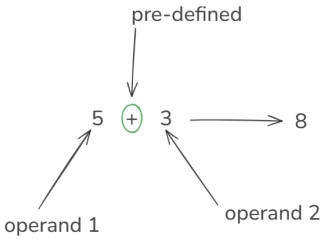


→ null is keyword

→ every class name in java is a non-primitive datatype

Primitive DataType	Non-Primitive DataType
1)To create Primtive Variables	1)To create Non-Primitive variables or Reference Variables
2)we have 8 Primitive Data Types	2) Infinite non-primitive datatypes are availabe
3)User defined is not possible	3)user defined is allowed

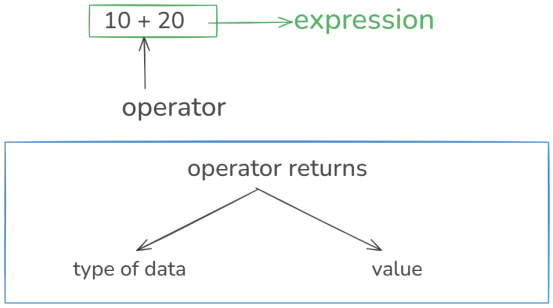
Operators :



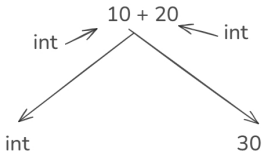
-->Operators are the pre-defined symbols which are used to perform specific task on the given operands(values).

Characteristics of an Operator:

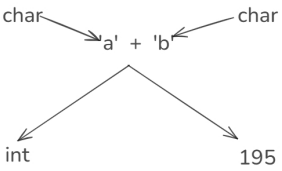
1) every operator returns the result / data after execution



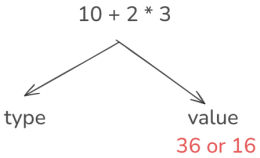
Ex 1 :



Ex2 :



2)Precedence : Priority given to the operators



→ pre-defined priority

→ whenever our expression contains more than one operator

( ) → 1st

/ * % → 2nd

+ - → 3rd

3)Associativity : any expression which is having more than one operator is complex

9 / 3 * 2

→ left to right

→ right to left

3 * 2

6

Ex : 10 + 'a' - 3 * 2

10 + 'a' - 6

107 -6

101

;

Ex : 'c' / 3 * 2

33 * 2

66

Types of Operators :

Operators

operands
-->Unary (performs operation on 1 data at a time)
-->Binary (accepts 2 values /data)
-->Ternary (accepts 3 values)

task
Arithemetic Operator
Relational Operator
Conditional Operator
Logical Operator
Increment and Decrement
Compound Assignment Operators

Arithemetic Operators:

-->All these operators are Binary Opeartors
Op1 + Op2

+
-
*
/
%

\+ → Addition Operator
\+ → Concat Operator

OP1 + OP2 → if any one of the operand is a String then it behave as Concat Operator

Ex :

1) 10 + 20 → Add -->30

2) "10" + 20 → concat → 1020
String int

3) 'a ' + 20 → Add
char int
97+20-->117

4) 'a' + "b" → ab (it will not take the asci value for concat)
char String

5) 10 + true → CTE
int boolean

6) "10" + true → Concat -->10true
String boolean

7) 10 + 20 + "dinga" → 30dinga
int int String

8) hello + 10 + 20 → CTE

* → it is used to perform the product of two numbers

Ex :

1) 10 * 2 → 20

2) 'a' * 2 →97*2-->194

3) "a" * 2 → CTE

4) true * 5 → CTE

/ → it performs division and gives the Quotient back

1) 10/2 →5

2)'a'/'b' →0 (1st value is < 2nd value) 1 (1st value is > 2nd value)

3) "a"/1 → CTE

→

Relational Operator: They are used to compare the values

→ Binary Operators(they accept 2 values)

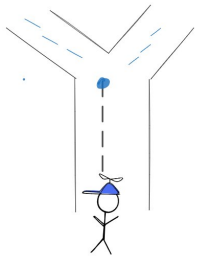
→ the return type of relational opearator is always Boolean

1) ==	True 10==10	True 'a'=='a'	False 5==4	True true==true
2) !=	False 10!=10	True 'a'!='b'	False true!=true	
3) >	False 10>20	True 20>10	False 'a'>'b'	
4) <	True 10<20	True 'a'<'b'		
5) >=	False 5>=10	True 10>=10	True 'k'>='k'	
6) <=	True 10<=10	False 10<=7	True 'a'<='b'	

Conditional Operators:

→ Ternary Operator

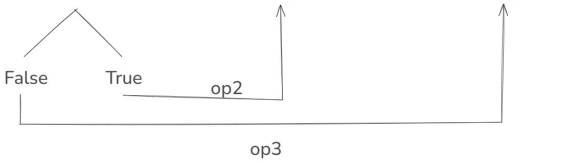
Syntax : OP1 ? OP2 : Op3



→ it behaves as Decision Making

OP1 --> Condition

Condition ? value/variable/expression : value/variable/expression



Ex : 10>20 ? 10 : 20

false

output : 20

Ex: int a=10;
int b=20;
System.out.println(a!=b ? a+b : a-b);

10!=20

true

a+b
10+20
30 output

→ int result = a!=b ? a+b : a-b ;

returns integer type of value

Ex :

String result = 'a'>='a' ? "Welcome..!" : "Tata Bye Byee...!" ;

97>=97

true

output : Welcome

Example Program

```
class Ternary
{
public static void main(String[] args)
{
System.out.println(10>20?10:20);

System.out.println('a'=='a'?10+20:20+30);

 int a=10;
 int b=5;
int result=a!=b?a*2:b*2;
System.out.println(result);

String result1='a'=='a'?"Welcome":"Bye Bye!";
System.out.println(result1);
}

}
```

Output

```
20
30
20
Welcome
```

Task:

Assume that we have 2 integer numbers stored in 2 variables
namely a and b ,write a java Program to obtain the Largest Number.

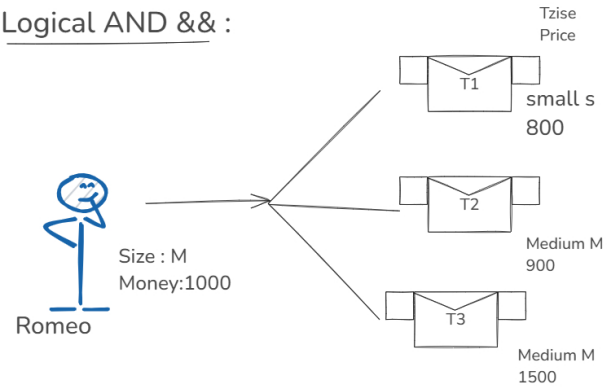
Logical Operator:

We have 3 types of Logical Operators

1)Logical AND &&
2)Logical OR ||
3)Logical NOT !

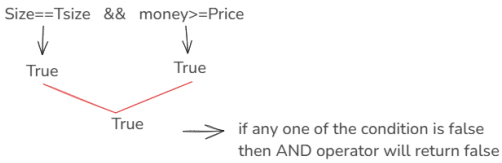
Binary
Unary

1)Logical AND && :



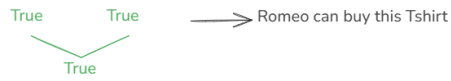
Cond1 -->Size==Tsize

Cond2 -->money>=Price

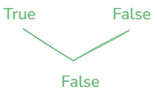


T1 :

T2: M==M && 1000>=900



T3: M==M && 1000>=1500

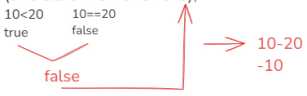


-->It is a Binary Operator
-->It works only on Boolean Data
-->return type is boolean(True or False)
-->We use AND operator only if both the conditions to be satisfied.

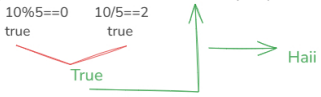
Truth Table for AND &&

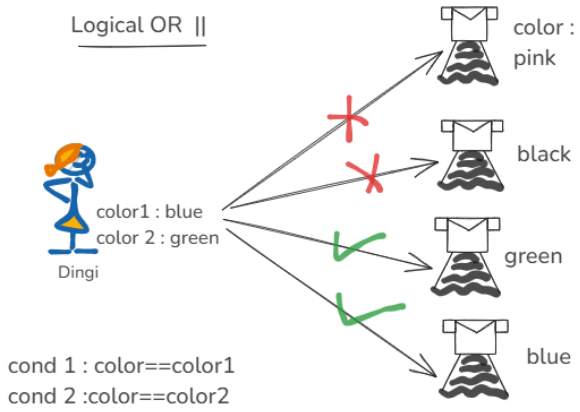
Cond1	Cond2	Result
false		false
true	false	false
true	true	true

Ex :
```
int a =10;
int b=20;
int result=(a<b && a==b ? a+b : a-b);
```



Ex :
```
int a=10;
int b=5;
System.out.println(a%b==0 && a/b==2 ? "Haii" : "Byee");
```





D1 color==color1 || color==color2
pink==blue || pink==green
False False
False

D2
Black==blue || black==green
False False
False

D3
green==blue || green==green
False True
True

D4
blue==blue || blue==green
True
True

-->It is a Binary Operator

-->it works only on boolean data
-->result is always boolean
-->we use OR || only when any one of the condition to be satisfied

Truth Table for OR ||

cond1	Cond2	Result
True		True
False	True	True
False	False	False

Ex :
10>5 || 10%5==0 ? "Hello" : "Byee";
True
True
output : Hello

Ex 2 :
'a'=='b' || 'a'>'b' ? 10+20 : 20+30 ;
False False
False
output : 50

int result= 'a'=='b' || 'a'>'b' ? 10+20 : 20+30 ;

Logical NOT ! :

-->Unary Operator
-->it can act on only one operand
-->it can work only on boolean data
-->it will Negate the boolean value

↓

Opposite

Syntax : `! Operand`

-->Operand can be a literal / variable / expression

Ex : ! true → false

! false → true

! 10 → CTE

! 10==10 → CTE

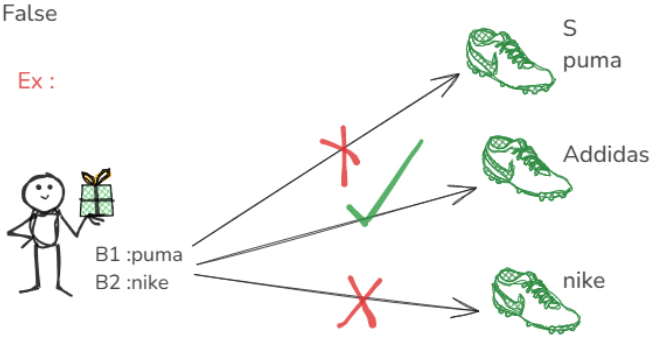
precedence

! (10==10)

↑ True

False

Ex :



Cond 1 : S==B1
cond 2: S==B2

S==B1 && S==B2

! (puma==puma && puma==nike)

False

True

S1 :

S==B1 || S==B2

! (puma==puma || puma==nike)

True False

True

False

S2:

! (addidas==puma || addidas==nike)

False False

False

→ He can Buy This Shoe :-)

True

S3 :

!(nike==puma || nike==nike)

False True

True

False

Increment Operator :

-->it is a unary operator
-->it is represented with " ++ "
-->it is used to update the variable by increasing 1

Increment(++)

post-increment
variable++

Ex : int a=10;

a 10 11

int b=a++;

b 10

System.out.println(a); //11
System.out.println(b); //10

-->Post Increment
variable ++
-->use the existing value inside the variable
-->then update the variable by increasing 1

pre-increment
++variable

Ex : int a=10;

a 10 11

int b = ++a;
11

b 11

System.out.println(a); //11
System.out.println(b); //11

-->Pre Increment
++ variable
-->First it will update the variable by increasing 1
-->then use the updated variable

Task : int a = 10 ;
int b=10;

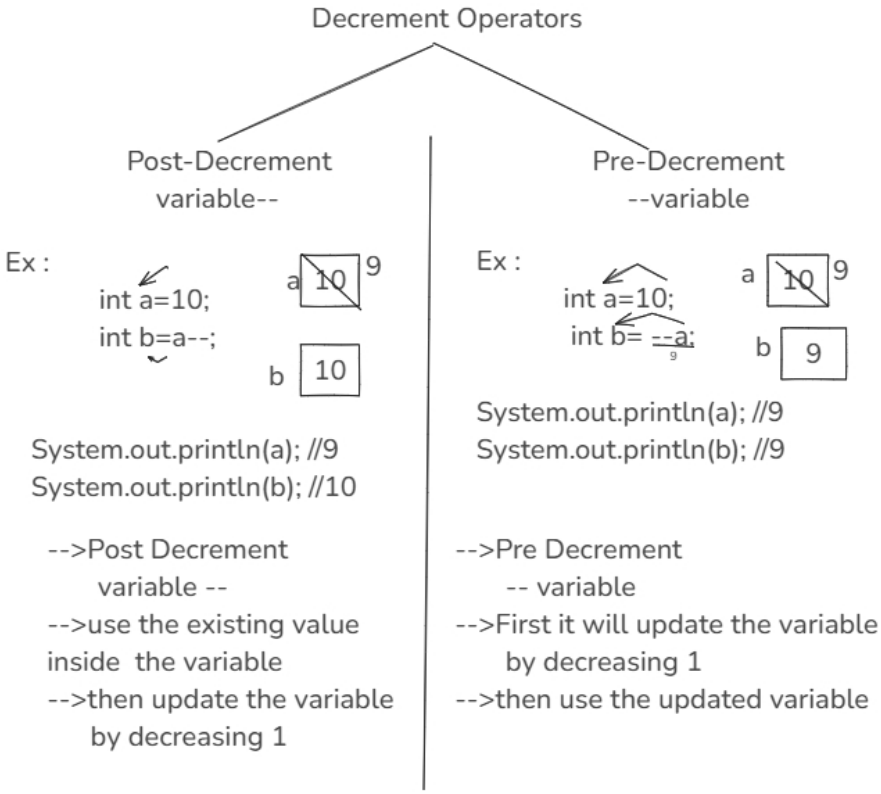
1) int c = a++ + ++b + ++b - ++a;
System.out.println(a);
System.out.println(b);
System.out.println(c);

2) int c= b++ - ++a + a++ + ++b;
System.out.println(a);
System.out.println(b);
System.out.println(c);

Increment Operators

Decrement Operator :

-->it is a unary operator
-->it is represented with " -- "
-->it is used to update the variable by decreasing 1



Note :

10 ++ → CTE

→ we can use this only with the variable

Task : int a=5;
int b=7;
1) int c = a-- + --b + --b - --a;
System.out.println(a);
System.out.println(b);
System.out.println(c);

2) int c= b-- - --a + a-- - --b;
System.out.println(a);
System.out.println(b);
System.out.println(c);

Decrement Operators

Task on Both Increment and Decrement:

int a=10;
int b=10;

1) int c= a++ + --b + b++ - --a
System.out.println(a);
System.out.println(b);
System.out.println(c);

2) int c= --b + ++a - ++b + ++a
System.out.println(a);
System.out.println(b);
System.out.println(c);

Compound Assignment Opeartor :

1) +=
2) -=
3) *=
4) /=
5) %=

variable += value/expression

+= → will add the new data with the existing data

varriable +=10;

variable = variable + 10;

Ex :

int a=10; a 10

a += 10; → a = a+10;

10+10

System.out.println(a);

//20

Ex : int a=5;

a*=5; → a=a*5

5*5

System.out.println(a);

//25

Ex :

Assume that there is 500 rs in dingas wallet
Dingi gave 700 rs to dinga and he put in his wallet
Now write a logic to print the total money present in his wallet.

Dinga

int wallet =500;

wallet+=700; → wallet=wallet+700;

500+700

1200

System.out.println(wallet);

//1200

Type Casting:

converting

type of data

-->it is the process of converting the value from one data type to another data type is known as Type Casting

10 → 10.0

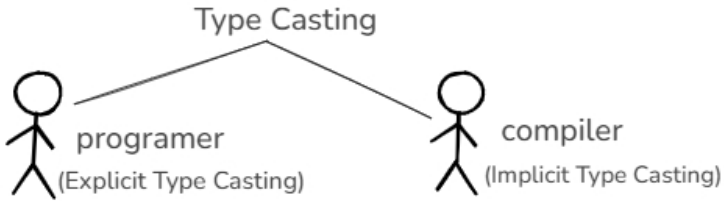
int double

convert the int data into double data

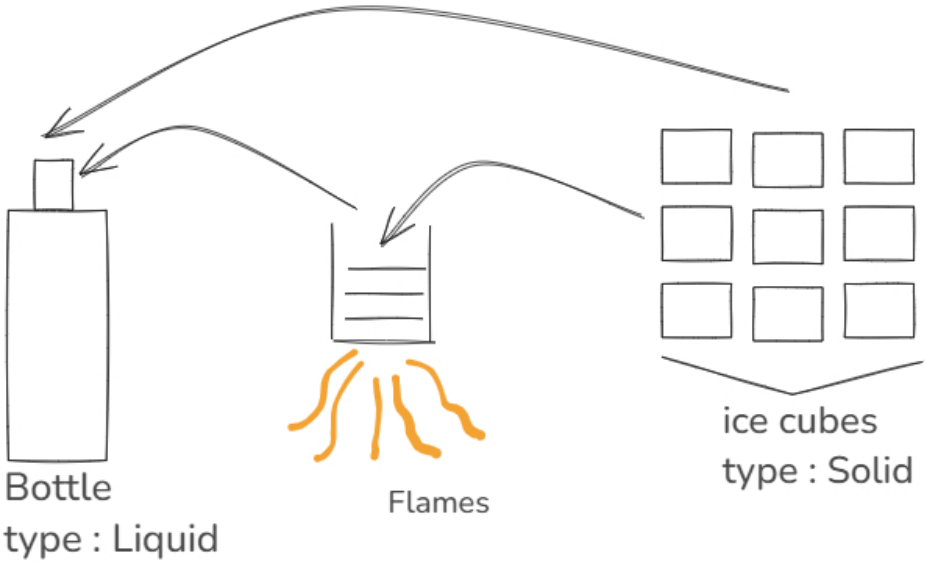
'A' → 65

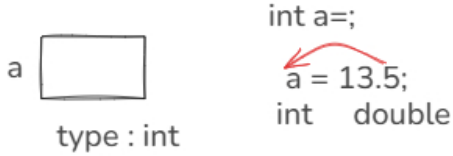
char int

Who can do the Type Casting???

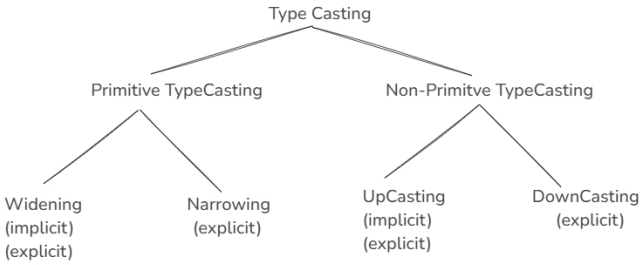


Why do we need Type Casting ??



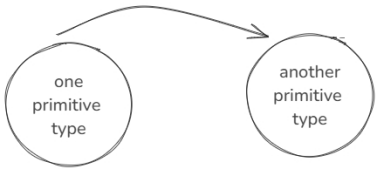


-->we cannot convert the data type but we can convert the value

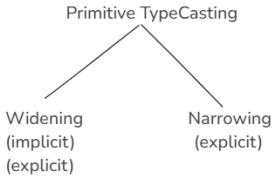


Primitive TypeCasting :

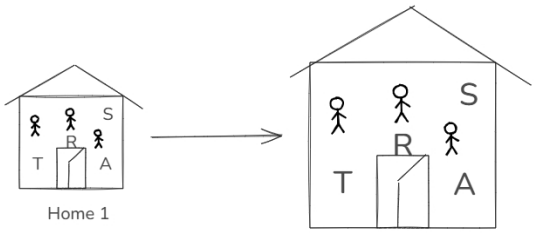
primitive types:
byte
short
int
long
double
float
char
boolean



-->the process of converting the value from one primitive type to another primitive type is known as Primitive Type Casting .



Widening :



no risk is involved
there is no loss

-->in java we are having some data types with increase in their capacity

byte < short < int < long < float < double

byte-->short
short-->int
int-->long

-->It is the process of converting the value from the smaller primitive type to the larger primitive type is known as Primitive TypeCasting

boolean → we cannot convert the value from any primitive type to boolean or boolean to any primitive type

char → int(widening)

Ex : int a = 10;

a | 10

double d;

d=a;

d | 10.0

10 → int

double

in this scenario the compiler will perform widening

```
System.out.println(a); // 10
System.out.println(d); //10.0
```

Ex : char c='a';

c | a

int b;

b = c;

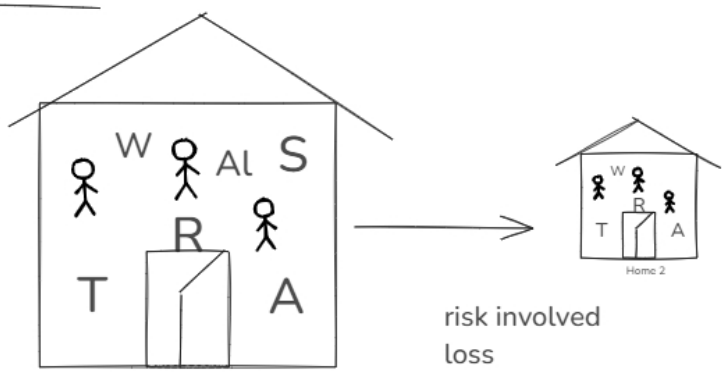
b | 97

int ← char

The compiler will perform the widening implicitly

```
System.out.println(c); //a
System.out.println(b); //97
```

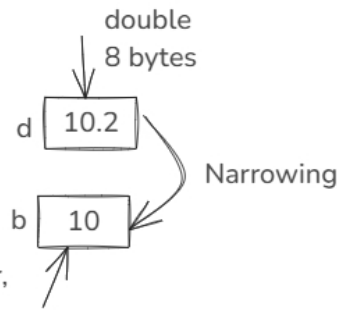
Narrowing :



risk involved
loss

Home 1

Ex : double d =10.2;
int b;
b = d;
10.2



-->Narrowing is never done by the Compiler,
Programer have to do it explicitly.

-->It is the process of converting the Value from Larger Primitive type to the Smaller Primitive type is known as Narrowing
-->Implicit type casting is not allowed

Ex : double → int
implicit ✗

byte < short < int < long < float < double

Explicit TypeCasting :

TypeCast Operator :

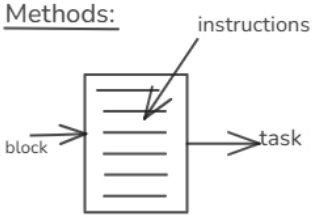
Syntax :
(type) value / variable / expression;
data to be converted

Ex : (int)10.2
10 ← double

Ex : long l=30;
int b;
b=(int)l;
30
int b=(int)l;

Ex : char c='a';
int b =(int)c;
forcefully doing the typecasting

Methods:



Method is a block of instructions which is used to perform
Specific task
Ex : Instagram page -->login page --> method will perform
action performed

-->for every action we can define multiple methods

Method Creation :

Syntax :

modifiers (optional) return-type name(Formal Arguments (optional))
{
return statement value/expression; (optional)
}

Sometimes it is mandatory,Some times it is optional

method declaration
method signature
modifiers return-type name(Formal Arguments)
{
return statement value/expression;
}
method block
method body
method implementation
method defination

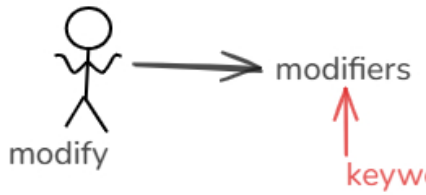
public static void main(String[ ] args)
{
modifiers
return type
method signature
method defination
}

-->we can create a method only inside a class

```
class ClassName
{
 public static void main(String[ ] args)
 {
  }
}
```

-->we can create any number of methods inside a class
-->In Java, we cannot create method inside method

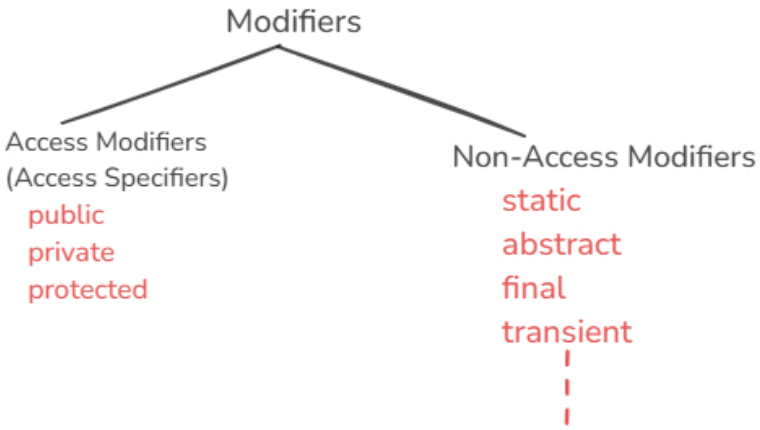
Modifiers :



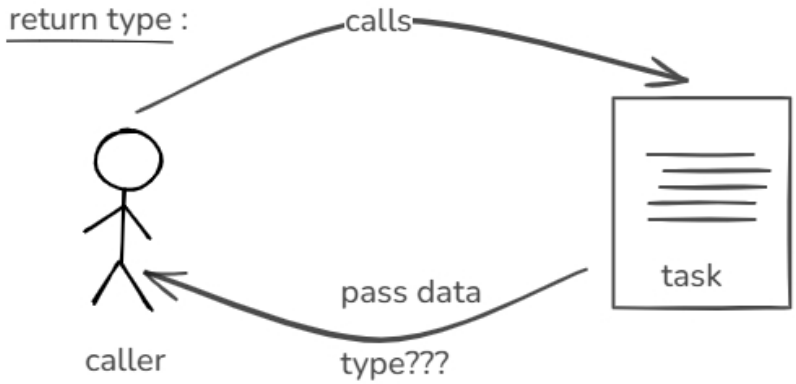
keywords

-->they are responsible to change the behaviour of the members of the class

-->some characteristics will be changed



return type :



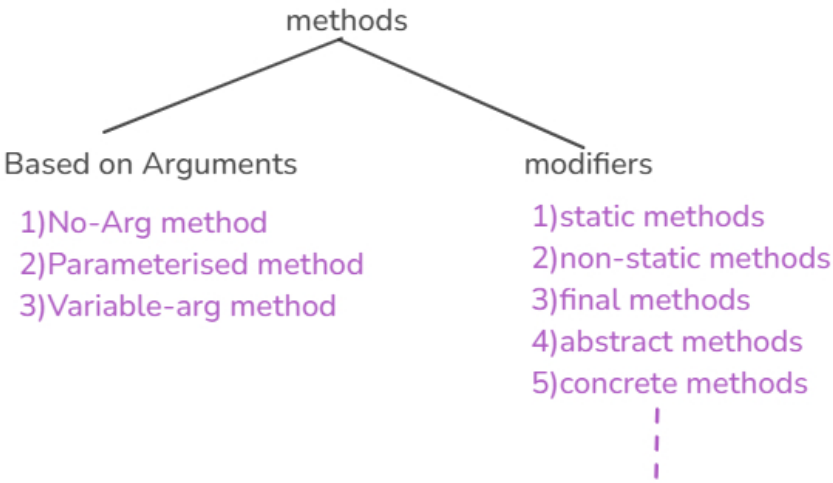
-->return type is a datatype , it tells what type of data is given back to the caller of the method once after the execution of method is completed

1)primitive type : byte,short,int,long,float,double,char,boolean

2)Non-primitive type : any classname,array

3) void : returns nothing

it is not returning any data back to the caller method



Characteristics of method:

1)method gets executed only when we call

```
class P1
{
public static void main(String[] args){
System.out.println("haii");
System.out.println("helloo");
}
public static void demo()
{
System.out.println("i am from demo");
}
}
```

not get executed
because it didnt get called

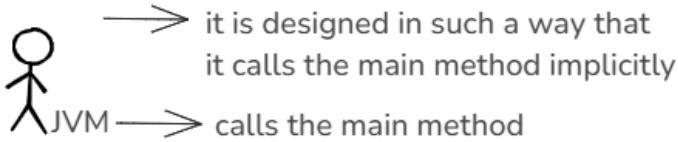
```
haii
helloo
```

2)methods can be called and executed any number of times
(Code Reusability)

```
class P1
{
public static void main(String[] args){
demo();
System.out.println("haiii");
demo();
System.out.println("byee");
demo();
}
public static void demo()
{
System.out.println("i am from demo");
}
}
```

```
C:\Users\ALPHA14\Desktop\javaprograms>javac P1.java

C:\Users\ALPHA14\Desktop\javaprograms>java P1
i am from demo
haiii
i am from demo
byee
i am from demo

C:\Users\ALPHA14\Desktop\javaprograms>
```

who calls main method :



1)No - Argument method :

A method created without formal Arguments is called No-Argument method

Syntax : modifiers return-type name( )

```
{

}
```

Advantages :

1)it is easy to call the method

```
name( )
```

DisAdvantage :

the caller of the method cannot pass the data to the method

* Design a method to perform addition of 2 numbers

caller

```
Ex : public static void main(String[] args)
     {
          add( );
       method call statement
          add(10,20);

     }
```

```
public static void add()
{
int a=10;
int b=20;
System.out.println(a+b); // 30
}
```

2)Parameterised method :

The methods which are having parameters/formal arguments is called as Parameterised method.

Syntax :

```
modifiers  return-type  name( datatype v1 , datatype v2 )
                                Formal Arguments
{

}
```

Formal Arguments : The variables created in the method declaration statement is called as Formal Arguments

Note :
1)Formal Arguments are local variables of the method

Advantage : caller can pass the data

Disadvantage : caller cannot call the method without passing the data

add( ); ✗

Syntax to call Parameterised method :

```
name(value/expression , value/expression);
```

Ex : design a method to print sum of 2 numbers

```
public static void main(String[] args)
{
   add(5,5);
   add( );
}
```

```
public static void add(int a,int b)
{
System.out.println(a+b);
}
```

Formal arguments

a and b are local variables to the method

Rules to call Parameterised method :

```
name(value/expression , value/expression);
      Actual Arguments
```

Rule 1 :

The number of Formal Arguments and Actual Arguments should be same.

Rule 2 :

The type of Formal Arguments and Actual Arguments should be same

Signature : demo(int)

demo ( ); ✗

demo(20); ✓

demo(10.5); ✗

demo('a'); ✓ Implicit Widening

demo(3,5); ✗

```
public static void demo(int a )
{


}
```

```
class B4
{
public static void main(String[] args)
{
add(5,7);

}
public static void add(int a,int b)
{
System.out.println(a+b);

}

}
```

```
C:\Users\ALPHA14\Desktop\javaprograms>javac B4.java

C:\Users\ALPHA14\Desktop\javaprograms>java B4
12

C:\Users\ALPHA14\Desktop\javaprograms>
```

✓

```
class B4
{
public static void main(String[] args)
{
add('a');

}
public static void add(int a)
{
System.out.println(a);

}

}
```

```
C:\Users\ALPHA14\Desktop\javaprograms>javac B4.java

C:\Users\ALPHA14\Desktop\javaprograms>java B4
97

C:\Users\ALPHA14\Desktop\javaprograms>
```

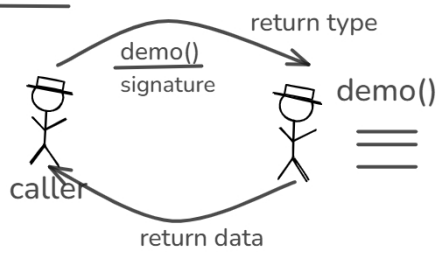
✓

```
class B4
{
public static void main(String[] args)
{
add(10);

}
public static void add(int a,int b)
{
System.out.println(a+b);

}

}
```

```
C:\Users\ALPHA14\Desktop\javaprograms>javac B4.java
B4.java:5: error: method add in class B4 cannot be applied to gi
ven types;
add(10);
^
  required: int,int
  found:    int
  reason: actual and formal argument lists differ in length
1 error

C:\Users\ALPHA14\Desktop\javaprograms>
```

✗

Method return :



-->we need to mention the return type when we declare a method
-->void - the method cannot return anything back to caller
-->if the method need to return anything back to the caller method then the return type should be other than void

Design a method which can return data to the caller :

--> there are 2 steps
Step 1 : provide the return type
Step 2 : use return statement with value/expression

Ex : design a method which can return sum of 2 integers

```
public static int add(int a,int b)
{
int result=a+b;
return result;
}
```

```
public static void main(String[] args)
{
    add(10,20);

}
```

```
public static int add(int a,int b)
{
System.out.println("hai");
return a+b;
}
```

To consume the data returned by the method we have 2 ways:

1)store the result inside a variable
2)use it as an expression inside any other statement

```
public static void main(String[] args)
{
   int res= add(10,20);
```

30

res 30

```
}
```

```
public static int add(int a,int b)
{
System.out.println("hai");
return a+b;
}
```

10+20

Return Statement :

return is a keyword

-->return is a control transfer statement , it stops the execution of current method and transfers the controller back to the caller

Syntax : return [value/variable/expression];

Rules :

1)return statement is mandatory,if the return type of the method is other than void

```
Ex :  int demo( )
      {          ------> CTE

      }
```

2)if the return type is void,return statement is optional

```
Ex : void demo( )
     {
     }

Ex : void demo( )
     {
      return 10;
     }

Ex : void demo( )
     {
      return;
     }
```

if the return type is void we should use the return statement without value

3)return statement in java , can return only one value

```
Ex : int demo ( )
     {
      return 10,20;
     }

Ex : int demo( )
     {
      return 10;
      return 20;
     }

Ex : int demo(int a )
     {
       return a;
     }
```

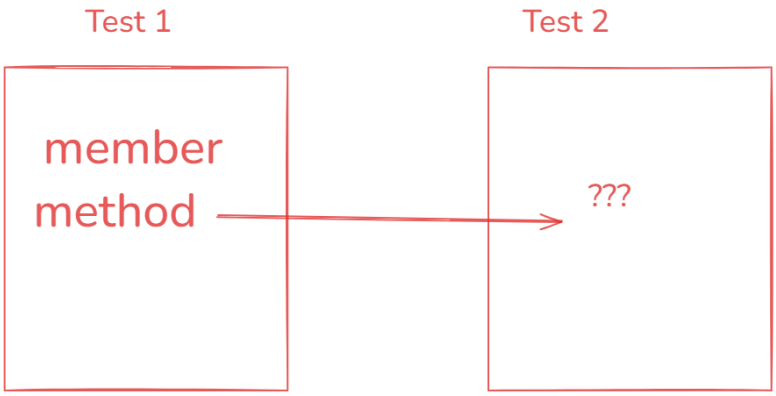
4) return statement should be the last statement of the block

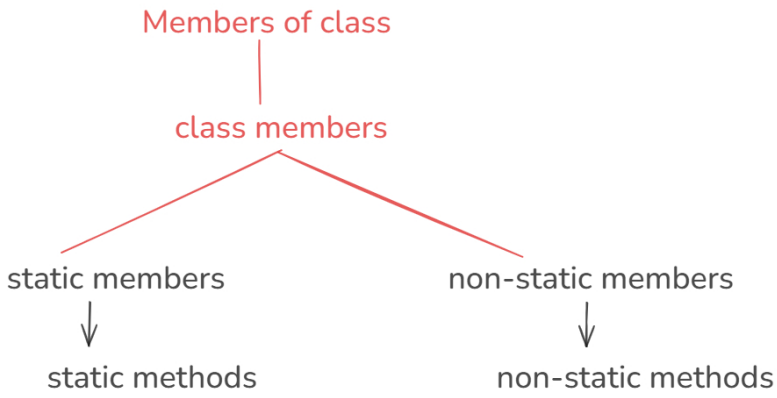
```
Ex : public static void demo()
     {
       System.out.println("hai");
         return;                          ------> syntatically wrong
       System.out.println("bye");
     }
```

Method as a member of a class

```
class Test1
{
public static int add(int a,int b)
{
return a+b;
}
public static void main(String[] args)
{
add(10,20);
}
}
```

add(int a,int b) is a member of Test1 class






```
class Demo
{
public static void demo( )
{

}
public void demo( )
{

}
```

static method

non-static method

consuming the data returned by method and storing it inside variable in method call statement

```
class K1
{
public static void main(String[] args)
{
int res=add(10,20);
System.out.println(res);
}
public static int add(int a,int b){
System.out.println("Hello ");
return a+b;

}



}
```

```
C:\Users\ALPHA14\Desktop\javaprograms>javac K1
va

C:\Users\ALPHA14\Desktop\javaprograms>java K1
Hello
30

C:\Users\ALPHA14\Desktop\javaprograms>
```

consuming the data in a Printing statement as an expression

```
class K1
{
public static void main(String[] args)
{
System.out.println(add(10,20));
}
public static int add(int a,int b){
System.out.println("Hello ");
return a+b;

}



}
```

```
C:\Users\ALPHA14\Desktop\javaprograms>java
1.java

C:\Users\ALPHA14\Desktop\javaprograms>java
Hello
30

C:\Users\ALPHA14\Desktop\javaprograms>
```

value/variable/expression in return statement is mandatory if your return type is other than void

```
class K1
{
public static void main(String[]
{
demo();
}
public static int demo()
{
System.out.println("hai");
return;
}

}
```

```
C:\Users\ALPHA14\Desktop\javaprograms>javac k1.java

C:\Users\ALPHA14\Desktop\javaprograms>java K1
hai

C:\Users\ALPHA14\Desktop\javaprograms>javac k1.java
k1.java:10: error: incompatible types: missing retu
rn value
return;
^
1 error

C:\Users\ALPHA14\Desktop\javaprograms>
```

you are not supposed to pass value/variable/expression in return statement if your return type is void

```
class K1
{
public static void main(String[] args)
{
demo();
}
public static void demo()
{
System.out.println("hai");
return 10;
}

}
```

```
C:\Users\ALPHA14\Desktop\javaprograms>javac K1.java
K1.java:10: error: incompatible types: unexpected return val
ue
return 10;
       ^
1 error

C:\Users\ALPHA14\Desktop\javaprograms>
```

return statement can return only one value

```
class K1
{
public static void main(String[] args)
{
demo();
}
public static int demo()
{
System.out.println("hai");
return 10;
return 20;
}

}
```

```
C:\Users\ALPHA14\Desktop\javaprograms>javac K1.java
K1.java:11: error: unreachable statement
return 20;
^
1 error

C:\Users\ALPHA14\Desktop\javaprograms>
```

we can cal a method inside any other method

```
class K1
{
public static void main(String[] args)
{
System.out.println("main starts");
demo();
System.out.println("main ends");

}
public static void demo()
{
System.out.println("i am from demo");
task();
System.out.println("demo ends");

}

public static void task()
{
System.out.println("i am from task");
}

}
```

```
C:\Users\ALPHA14\Desktop\javaprograms>javac K1.jav
a

C:\Users\ALPHA14\Desktop\javaprograms>java K1
main starts
i am from demo
i am from task
demo ends
main ends

C:\Users\ALPHA14\Desktop\javaprograms>
```

to use static method of a class inside the same class :

```
class Test1{
public static void main(String[] args){
 demo();
 Test1.demo();
}
public static void demo(){
System.out.println("i am from demo");
}
}
```

--> 2 ways
1) by method name directly
2) classname

to use static method of a class inside the different class :

```
class Test1 {
public static void demo(){
System.out.println("helloo");
}
}
```

```
class Test2
{
public static void main(String[] args)
{
demo();
Test1.demo();
}
}
```

To declare a non-static method :

```
class Test
{
public void demo()
{
System.out.println("haiiiii");
}
}
```

non-static method

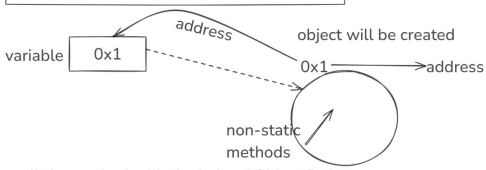
-->a method without static modifier is known as non-static method

```
class Test{
public void demo()
{
System.out.println("Haiiiii");
}
public static void main(String[] args)
{
 demo();
 Test.demo();
}
}
```

Steps to execute a non-static method :

Step 1 : create an Instance(Object) for the class

```
ClassName  variable = new ClassName( );
```



Step 2 : call the method with the help of Object Reference

```
reference_variable.methodcall( );
```

Ex :

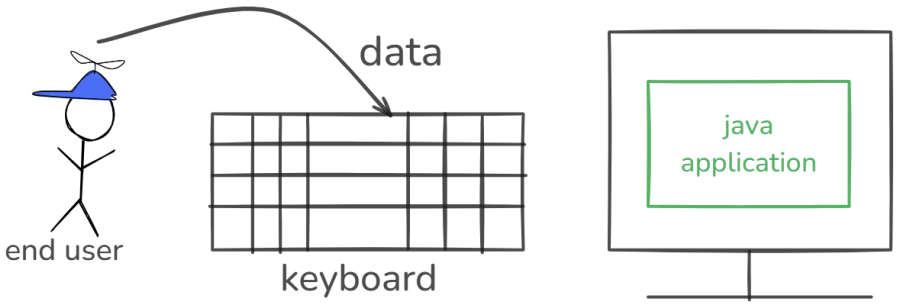
```
class Test{
public void demo()
{
System.out.println("Haiiiii");
}
public static void main(String[] args)
{
  Test  t =new Test( );
     t.demo( );
}
}
```

Ex :

```
class Study
{
public int add(int n1,int n2)
 {
   return n1+n2;
 }
}
```

```
class Study2
{
public static void main(String[] args)
{
//addition of 2 numbers
 //create an object for study class bcz non-static method
//is present inside that
 Study s=new Study( );
  int result=s.add(10,20);
System.out.println(result);
System.out.println(s.add(5,5));

}
}
```

DYNAMIC READ :



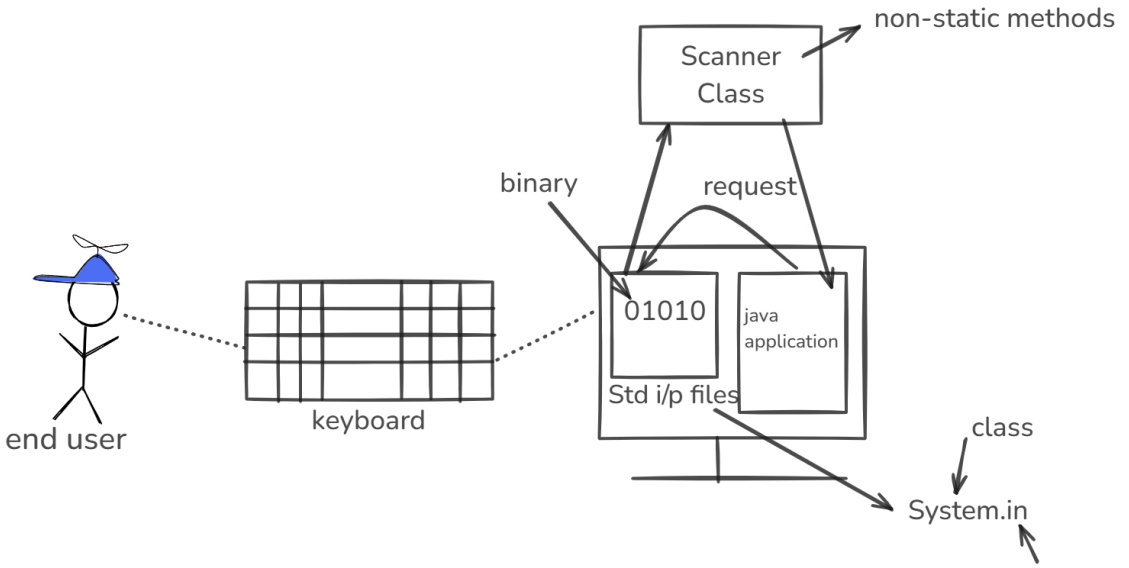
-->Fetching the data from the enduser when the application is under the execution is known as Dynamic Read

-->In java we can achieve Dynamic read with the help of Scanner Class

↓

java.util package

SCANNER CLASS :



-->Buffer files are also called as Standard Input Files

To use methods of java.util.Scanner :

Step 1 : import the Scanner Class from java.util

```
import java.util.Scanner;
```

Step 2 : Create Object for the Scanner class and intiliaze it with System.in

```
Scanner s=new Scanner(System.in);
```

Step 3 : request for data by calling

```
s.nextInt( );
s.nextShort( );
s.nextByte( );
s.nextDouble( );
s.nextLong( );
s.nextBoolean( );
s.nextFloat( );
```

Hello world

```
s.next( ); -->return a string one word or upto sapce
s.nextLine( );-->string
s.next( ).charAt(0); -->char
```

Method OverLoading :

```
class Demo{
public static void main(String[] args){
System.out.println("hello");
}
public static void demo( ){
System.out.println("hello everyone");
}

public static void demo( ){
System.out.println("hai everyone");
}

public static void demo(int a){
System.out.println("byee everyone");
}
```

Def : Having one more than one method with same name but different in method signatures is known as Method Overloading

-->Access Modifiers can be same or different
-->Access Specifiers can be same or different
-->return type can be same or different

-->Formal Arguments should be different in the method signature

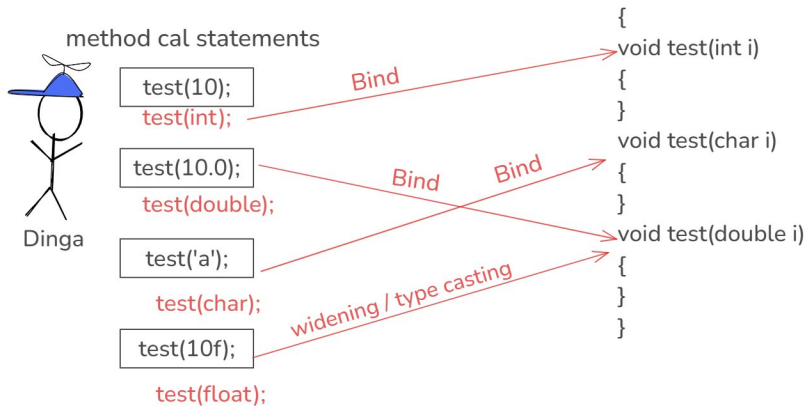
Rules :

1)Name of the method should be same
2)List of Arguments should be differ by either length or by type

```
Ex : class P1{
   void test (int i)
  {              1 argument

   }
void test ( byte b)
{                 1 argument but
                  type is different
}
void test(int i,int j)
{                 2 arguments
                  (different in length)
}
}
```

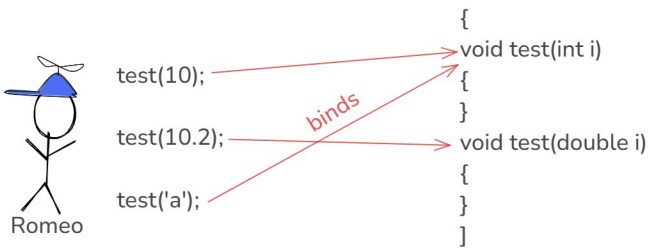
Note : return type and modifiers can be anything or same

Compile Time Binding / static Binding :

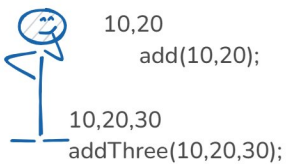


Def : The process of Binding the Method Call Statement with the method Defination during the compile time by the compiler by analysing or looking at actual arguments and formal arguements is known as Compile Time Binding.

-->compiler always binds with same signature ,if it is not present then it try for type promotion



Case 1 : without method overloading

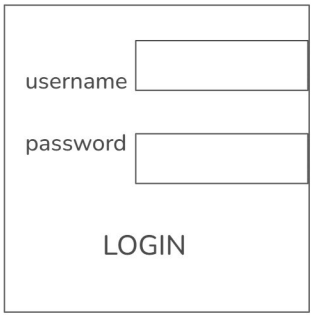


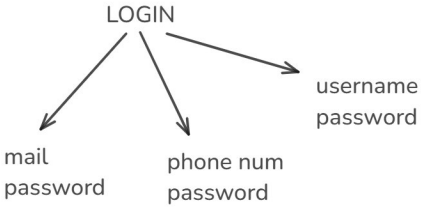

```
class Calci{
p s int add(int a ,int b)
{
return a+b;
}

p s int addThree(int a,int b,
                 int c)
{
return a+b+c;
}
}
```

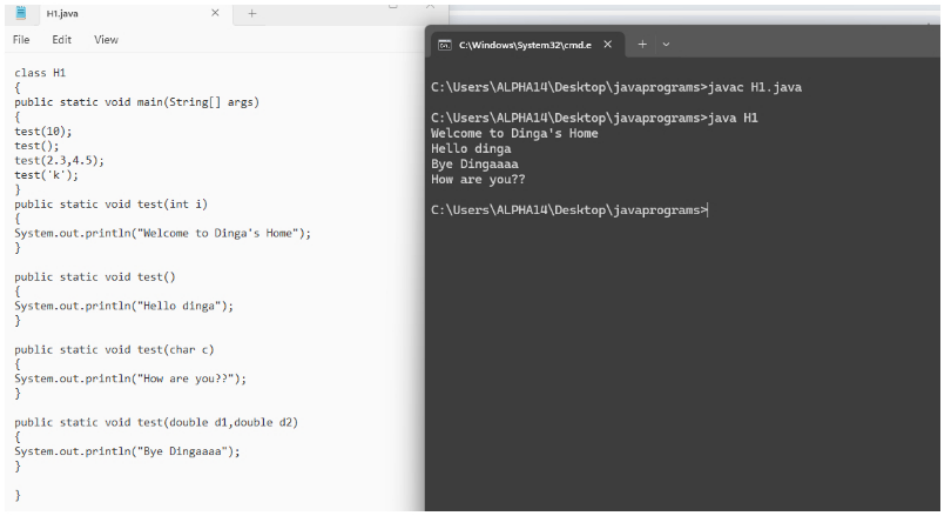
Case 2 : with method OverLoading

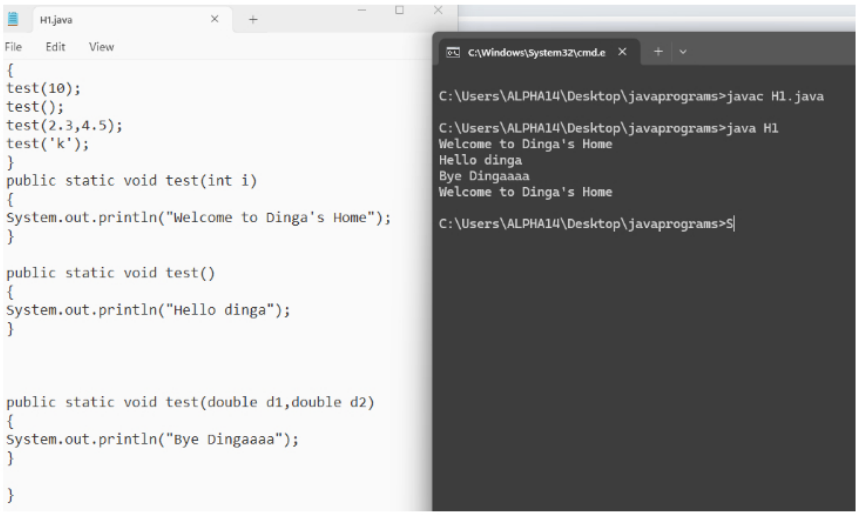

```
class Calci{
p s int add(int a ,int b)
{
return a+b;
}

p s int add(int a,int b, int c)
{
return a+b+c;
}
}
```

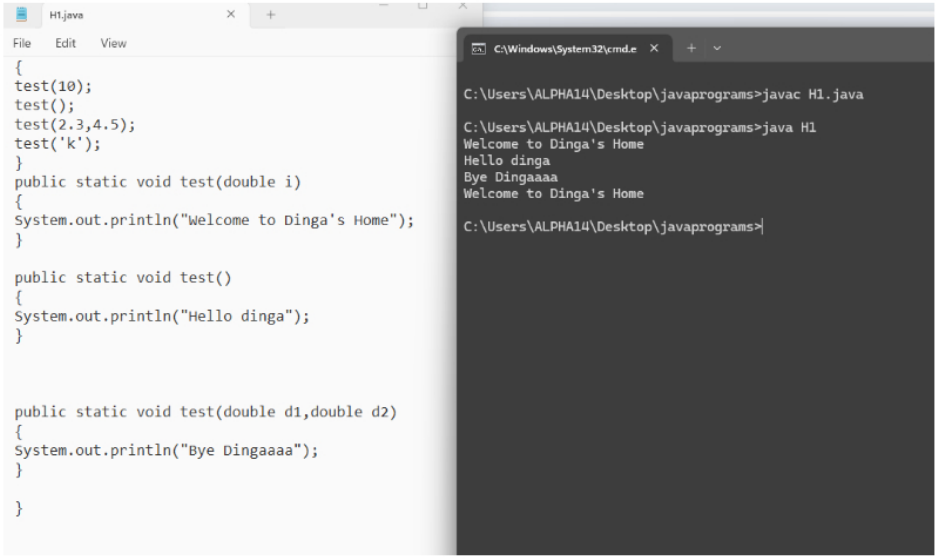
Instagram

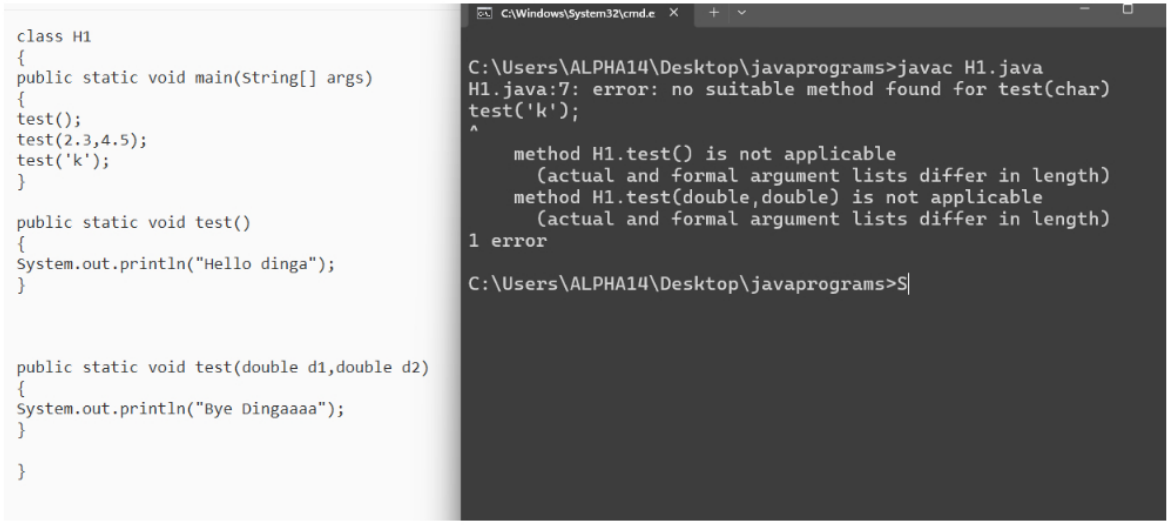





H1.java
File Edit View
class H1
{
public static void main(String[] args)
{
test(10);
test();
test(2.3,4.5);
test('k');
}
public static void test(int i)
{
System.out.println("Welcome to Dinga's Home");
}
public static void test()
{
System.out.println("Hello dinga");
}
public static void test(char c)
{
System.out.println("How are you??");
}
public static void test(double d1,double d2)
{
System.out.println("Bye Dingaaaa");
}
}
C:\Windows\System32\cmd.e
C:\Users\ALPHA14\Desktop\javaprograms>javac H1.java
C:\Users\ALPHA14\Desktop\javaprograms>java H1
Welcome to Dinga's Home
Hello dinga
Bye Dingaaaa
How are you??
C:\Users\ALPHA14\Desktop\javaprograms>


H1.java
File Edit View
{
test(10);
test();
test(2.3,4.5);
test('k');
}
public static void test(int i)
{
System.out.println("Welcome to Dinga's Home");
}
public static void test()
{
System.out.println("Hello dinga");
}
public static void test(double d1,double d2)
{
System.out.println("Bye Dingaaaa");
}
}
C:\Windows\System32\cmd.e
C:\Users\ALPHA14\Desktop\javaprograms>javac H1.java
C:\Users\ALPHA14\Desktop\javaprograms>java H1
Welcome to Dinga's Home
Hello dinga
Bye Dingaaaa
Welcome to Dinga's Home
C:\Users\ALPHA14\Desktop\javaprograms>S


H1.java
File Edit View
{
test(10);
test();
test(2.3,4.5);
test('k');
}
public static void test(double i)
{
System.out.println("Welcome to Dinga's Home");
}
public static void test()
{
System.out.println("Hello dinga");
}
public static void test(double d1,double d2)
{
System.out.println("Bye Dingaaaa");
}
}
C:\Windows\System32\cmd.e
C:\Users\ALPHA14\Desktop\javaprograms>javac H1.java
C:\Users\ALPHA14\Desktop\javaprograms>java H1
Welcome to Dinga's Home
Hello dinga
Bye Dingaaaa
Welcome to Dinga's Home
C:\Users\ALPHA14\Desktop\javaprograms>


class H1
{
public static void main(String[] args)
{
test();
test(2.3,4.5);
test('k');
}
public static void test()
{
System.out.println("Hello dinga");
}
public static void test(double d1,double d2)
{
System.out.println("Bye Dingaaaa");
}
}
C:\Windows\System32\cmd.e
C:\Users\ALPHA14\Desktop\javaprograms>javac H1.java
H1.java:7: error: no suitable method found for test(char)
test('k');
^
method H1.test() is not applicable
(actual and formal argument lists differ in length)
method H1.test(double,double) is not applicable
(actual and formal argument lists differ in length)
1 error
C:\Users\ALPHA14\Desktop\javaprograms>S

We can cal a static method inside static context directly or using ClassName

```
File  Edit  View

class Test1{
public static void main(String[] args){
demo();
Test1.demo();
}
public static void demo()
{
System.out.println("Helloooo");
}
}
```

```
C:\Users\ALPHA14\Desktop\javaprograms>javac Test1.java

C:\Users\ALPHA14\Desktop\javaprograms>java Test1
Helloooo
Helloooo

C:\Users\ALPHA14\Desktop\javaprograms>
```

if you try to cal the static method inside another class directly it throws error

```
File  Edit  View

class Test2
{
public static void main(String[] args)
{
demo();
}
}
```

```
C:\Windows\System32\cmd.e

C:\Users\ALPHA14\Desktop\javaprograms>javac Test2.java
Test2.java:5: error: cannot find symbol
demo();
^
  symbol:   method demo()
  location: class Test2
1 error

C:\Users\ALPHA14\Desktop\javaprograms>
```

we need to call a static method inside another class with the help of a ClassName

```
class Test2
{
public static void main(String[] args)
{
Test1.demo();
}
}
```

```
C:\Users\ALPHA14\Desktop\javaprograms>javac Test2.java

C:\Users\ALPHA14\Desktop\javaprograms>java Test2
Helloooo

C:\Users\ALPHA14\Desktop\javaprograms>
```

to call a non-static method inside the static method in the same class

```
class Test
{
public void demo()
{
System.out.println("Haiiiiiiii");
}
public static void main(String[] args)
{
Test t=new Test();
t.demo();
}
}
```

```
C:\Users\ALPHA14\Desktop\javaprograms>javac Test.java

C:\Users\ALPHA14\Desktop\javaprograms>java Test
Haiiiiiiii

C:\Users\ALPHA14\Desktop\javaprograms>
```

to call a non-static method inside the static method of different class

```
File  Edit  View

class Study2{
public static void main(String[] args)
{
Study s=new Study();
int result=s.add(10,20);
System.out.println(result);
System.out.println(s.add(5,3));
}
}
```

```
C:\Users\ALPHA14\Desktop\javaprograms>javac Study2.java

C:\Users\ALPHA14\Desktop\javaprograms>java Study2
30
8

C:\Users\ALPHA14\Desktop\javaprograms>
```

Using Scanner Class

```
import java.util.Scanner;
class Hello
{
public static void main(String[] args)
{
System.out.println("welcome");
Scanner s=new Scanner(System.in);
System.out.println("Enter the value for a : ");
int a=s.nextInt();
System.out.println("Enter the value for b : ");
int b=s.nextInt();
System.out.println(a+b);
System.out.println("Thank you");
}
}
```

```
C:\Users\ALPHA14\Desktop\javaprograms>javac Hello.java

C:\Users\ALPHA14\Desktop\javaprograms>java Hello
welcome
Enter the value for a :
10
Enter the value for b :
5
15
Thank you

C:\Users\ALPHA14\Desktop\javaprograms>javac Hello.java

C:\Users\ALPHA14\Desktop\javaprograms>java Hello
welcome
Enter the value for a :
5
Enter the value for b :
7
12
Thank you
```

Decision Statements :

Ex : Instagram

Step 1 : Open instagram
Step 2: Enter UserName
Step 3:Enter Password
Step 4 :click on login

o/p : user Home Page

class Login {

public static void main(String[] args){
S1 : read the user name
S2 : read the password
S3:Display the Home page

Rocky@gmail.com
Rocky123
Rocky12

-->It helps the programmer to decide whether an instruction or block of instructions to be executed or skipped

Types of Decision Statements :

1) if
2) if else
3) else if ladder
4) switch

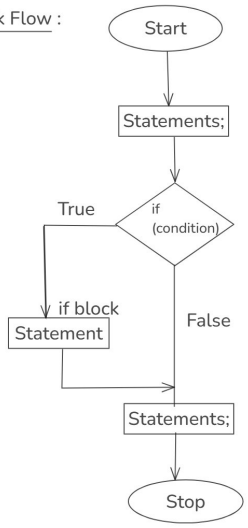
1) if : it is a keyword which is used as a Decision statement

Syntax :

```
if(condition)
  statement;          ---> to execute one statement

if(condition)
{
statements;           ---> to execute multiple statements
    |
    |
}
```

-->if the condition is true , it will execute or else it skips the execution of if block

Work Flow :



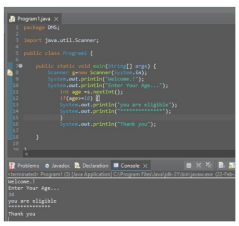
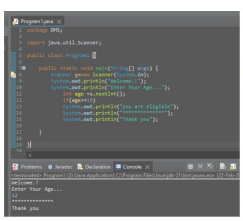
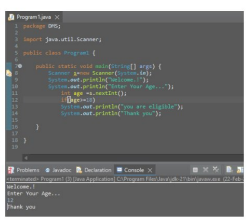
Ex : check whether the candidate is eligible to vote or not

case 1 : 14

```
Welcome
Enter your age
_
Thank you
```

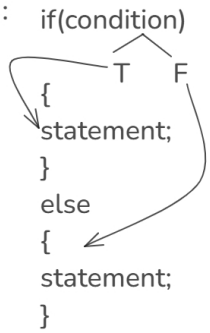
case 2 : 23

```
Welcome
Enter your age
_
you are eligible
Thank You
```

2) if else :

Syntax :

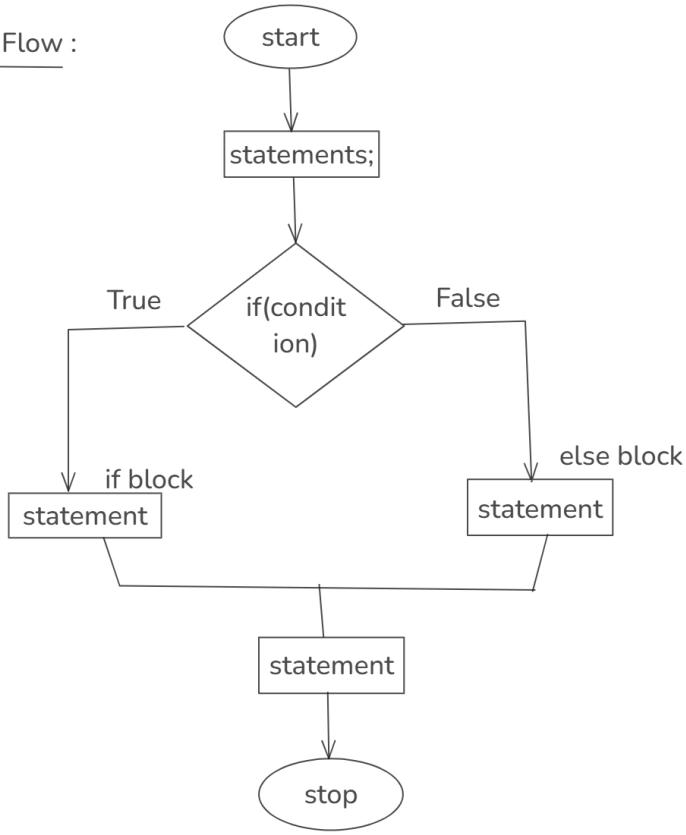


-->for one statement block is not neccesary

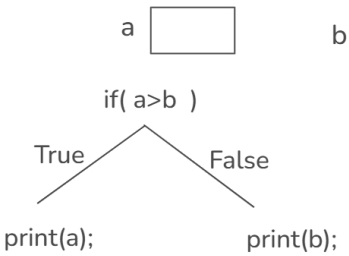
-->if condition is true , only if block gets executed and else block will not be executed

-->if condition is false then else block gets executed

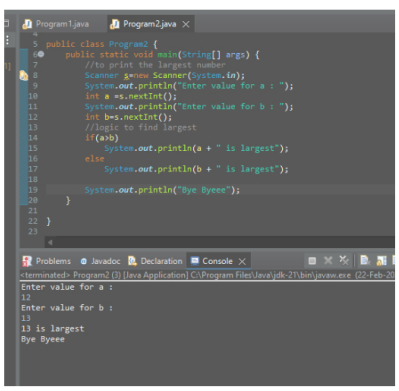
Work Flow :

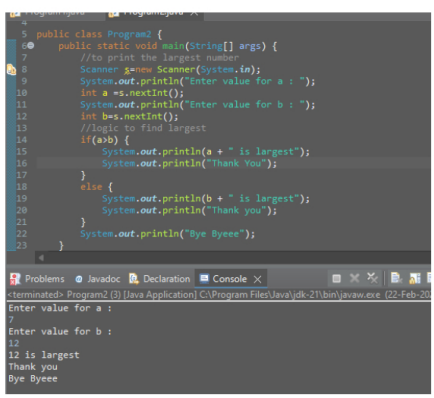


Ex : Assume we have 2 integer numbers stored in containers a and b.Write a logic to print the Largest number




```
if(a>b)
{
S.o.pn(a);
}
else
{
S.o.pl(b);
}
```



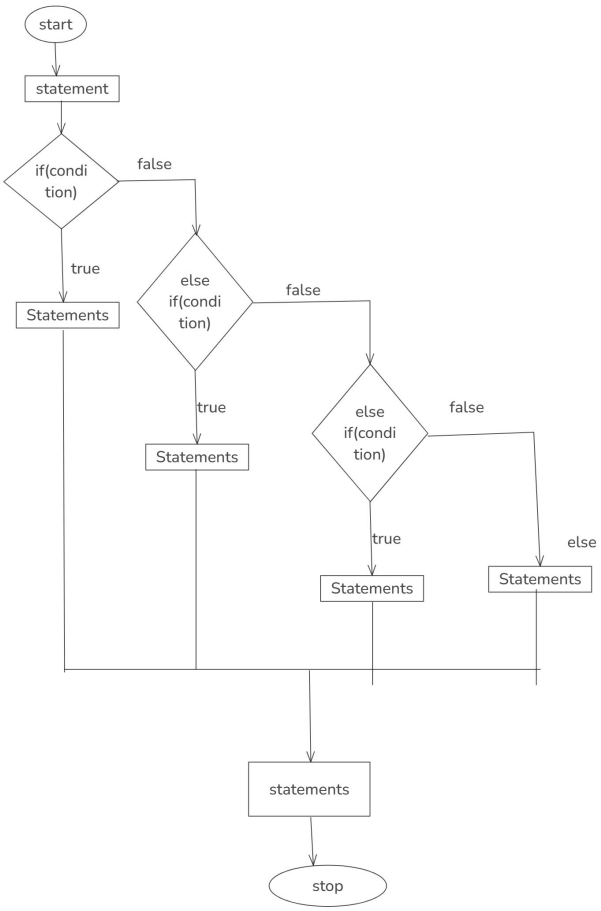


3) else if :

Syntax :
```
if (condition)
{
 statements;
 }
 else if(condition)
 {
  statements;
  }
  else if(condition)
    {
     statements;
      }
   else
  {
 statements;
  }
```

Rules : out of these only one block can be executed
-->else block should be written at the last
-->else block is optional
-->if the 1st condition is true,it will never go to the next condition
-->

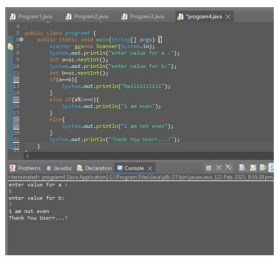
Work Flow :



Ex : Take 2 integer numbers store inside a and b
-->check whether both a and b are equal or not
-->if equal print Haiii
-->if not equal then check whether a is even or not
if even printi am even ,if not print i am odd

a [ ] b [ ]

```
if(a==b)
{
S.o.pn("Haiiii");
}
else if(a%2==0)
{
S.o.pn("I am even");
}
else
{
S.o.pn("i am odd");
}
```



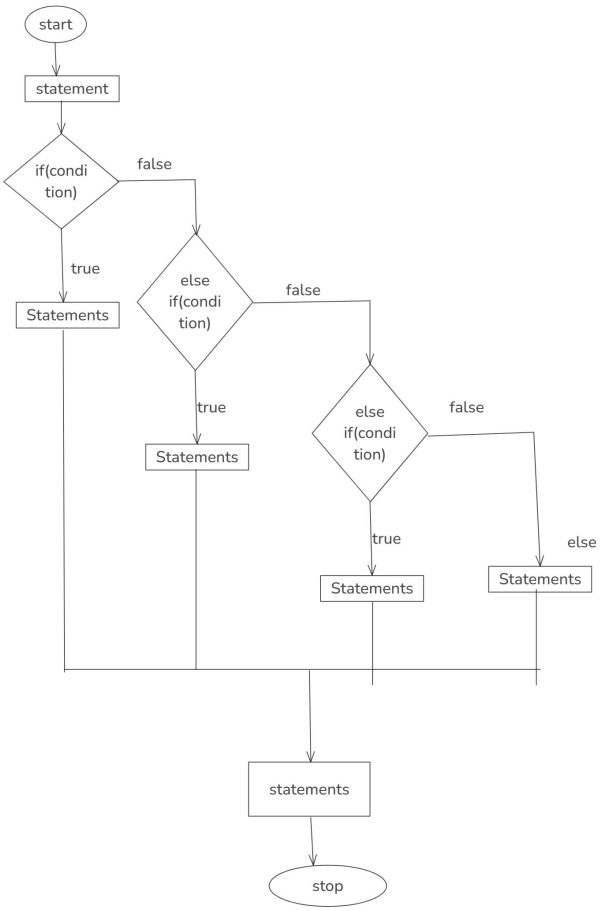
Task : Take 2 integer numbers store inside a and b
-->check whether both a and b are equal or not
-->if equal print we are equal
-->if not equal then print the largest.

3) else if :

Syntax :
```
if (condition)
{
 statements;
 }
 else if(condition)
 {
  statements;
  }
  else if(condition)
    {
     statements;
      }
   else
  {
 statements;
  }
```

Rules : out of these only one block can be executed
-->else block should be written at the last
-->else block is optional
-->if the 1st condition is true,it will never go to the next condition
-->

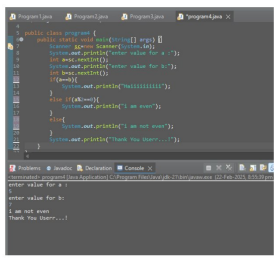
Work Flow :



Ex : Take 2 integer numbers store inside a and b
-->check whether both a and b are equal or not
-->if equal print Haiii
-->if not equal then check whether a is even or not
if even printi am even ,if not print i am odd

a [ ] b [ ]

```
if(a==b)
{
S.o.pn("Haiiii");
}
else if(a%2==0)
{
S.o.pn("I am even");
}
else
{
S.o.pn("i am odd");
}
```



Task : Take 2 integer numbers store inside a and b
-->check whether both a and b are equal or not
-->if equal print we are equal
-->if not equal then print the largest.

4)switch :

Syntax :

```
switch(value/variable/expression)
{
case value/exp :
        {
          statements;
         break;
         }
case value/exp :
        {
          statements;
         break;
        }
default :
       {
        statements;
        }

}
```

entire default block is optional
break is also optional

Rules :

1)byte,short,int,char,String are allowed in value / expression
2)in cases constants are allowed

Ex : 1 ✓
1+2 ✓
a ✗
a+2 ✗
'a' ✓

Ex :

```
int a=2;
switch(a) 2
{
case 1 : S.o.pln("hai");
case 2 : S.o.pln("hello");
case 3 :S.o.pln("namasthe");
default : S.o.pln("Bye");
}
```

```
hello
namasthe
bye
```

```
int a=2;
switch(a) 2
{
case 1 : S.o.pln("hai");
          break;
case 2 : S.o.pln("hello");
          break;
case 3 :S.o.pln("namasthe");
         break;
default : S.o.pln("Bye");
}
```

```
hello
```

Ex : String s="Dosa";

```
switch(s)
{
case  "Idli" :
        {
          S.o.pln("idli is on monday");
          S.o.pln("idly with sambar is too good");
         }
          break;
case "puri" :
       {
          S.o.pln("puri is on tuesday");
          S.o.pln("puri with aalu kurma is too tasty");
         }
          break;
case  "Dosa" :
          {
          S.o.pln("Dosa  is on wednesday");
          S.o.pln("Dosa is so yummmm");
         }
         break;
default :
       {
         S.o.pln("enter a valid tiffin");
          S.o.pln("***************");
         }
```

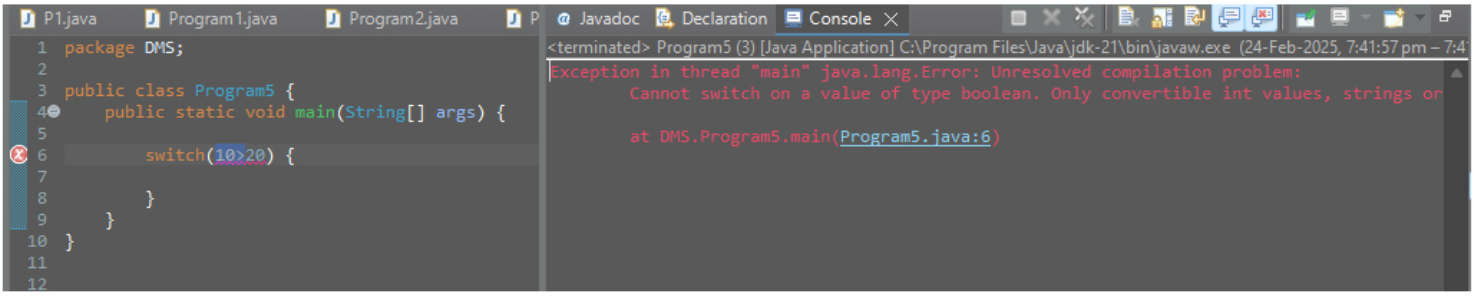
```
Dosa is on wednesday
Dosa is so yummmmm
```

without break :

---enter from i to m ------

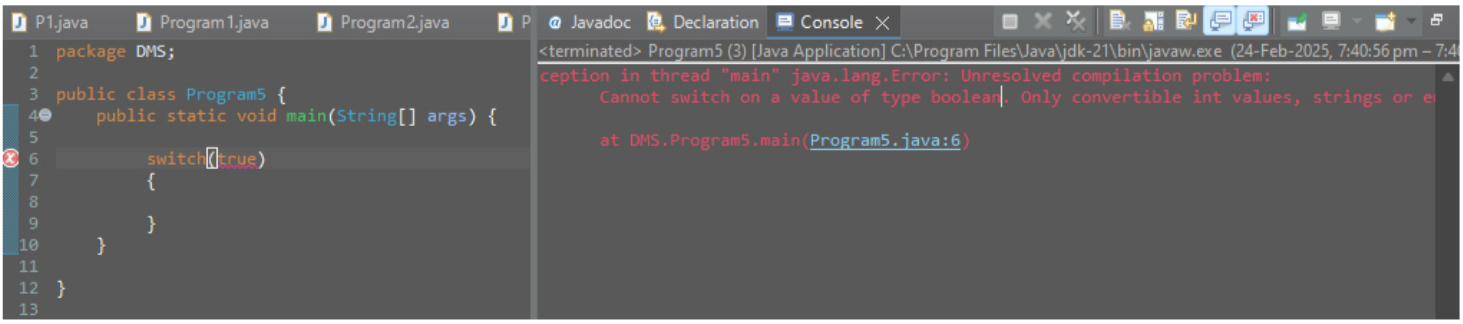
```
switch('k')
{
case 'i' : S.o.pln(" I for Ishq");
case 'j' : S.o.pln("j for jawan");
case 'k' : S.o.pln("k for KGF");
case 'l' : S.o.pln(" l for lagan");
case 'm' : S.o.pln("m for munna bhai");
           break;
default : S.o.pln("enter a valid character ");
}
```

```
k for KGF
l for lagan
m for munna bhai
```



```
package DMS;

public class Program5 {
    public static void main(String[] args) {

        switch(10>20) {

        }
    }
}
```

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
        Cannot switch on a value of type boolean. Only convertible int values, strings or

        at DMS.Program5.main(Program5.java:6)
```

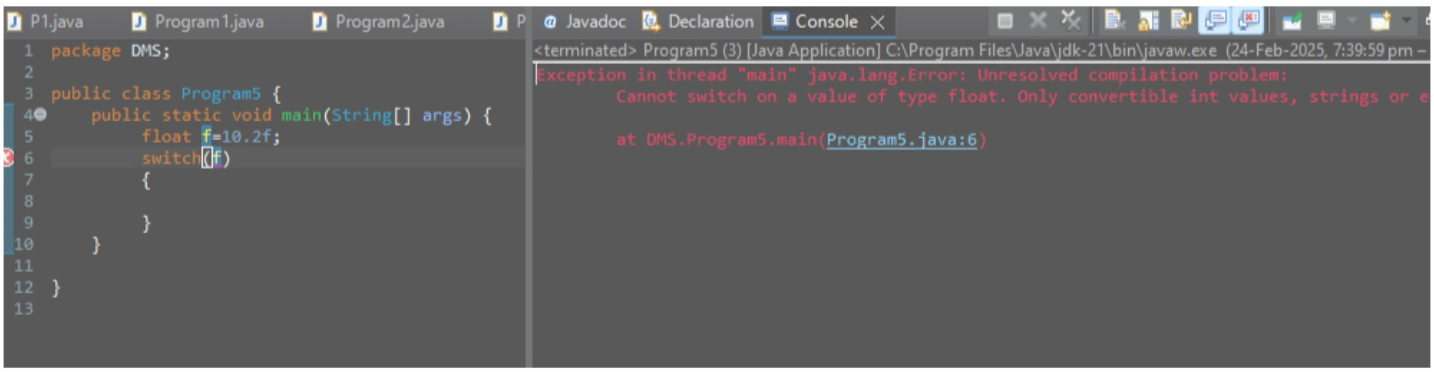


```
package DMS;

public class Program5 {
    public static void main(String[] args) {

        switch(true)
        {

        }
    }

}
```

```
ception in thread "main" java.lang.Error: Unresolved compilation problem:
      Cannot switch on a value of type boolean. Only convertible int values, strings or e

      at DMS.Program5.main(Program5.java:6)
```

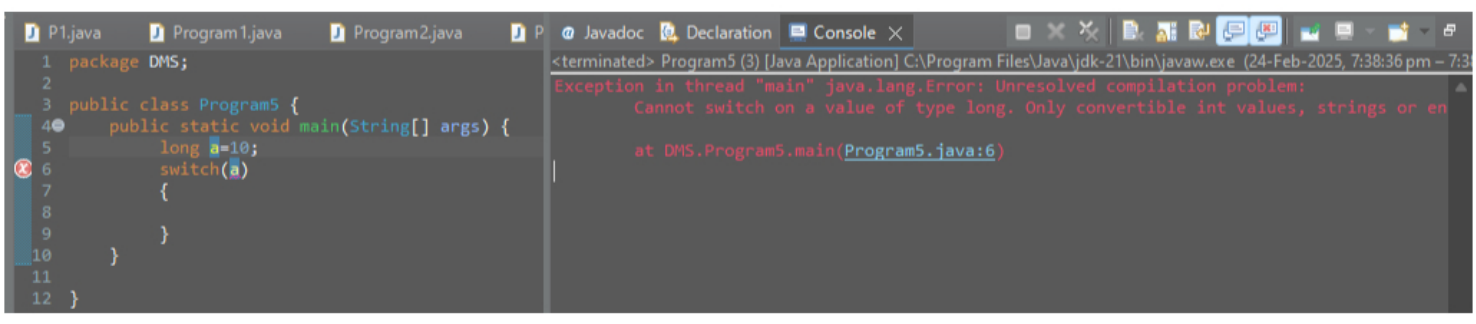
```
package DMS;

public class Program5 {
    public static void main(String[] args) {

        switch(10.2f)
        {

        }
    }

}
```

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
        Cannot switch on a value of type float. Only convertible int values, strings or e

        at DMS.Program5.main(Program5.java:6)
```



```
package DMS;

public class Program5 {
    public static void main(String[] args) {
        float f=10.2f;
        switch(f)
        {

        }
    }

}
```

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
        Cannot switch on a value of type float. Only convertible int values, strings or e

        at DMS.Program5.main(Program5.java:6)
```



```
package DMS;

public class Program5 {
    public static void main(String[] args) {
        long a=10;
        switch(a)
        {

        }
    }

}
```

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
        Cannot switch on a value of type long. Only convertible int values, strings or en

        at DMS.Program5.main(Program5.java:6)
```

without using break

```java
public class Program6 {
public static void main(String[] args) {
    //to printall the bitting in a week
    Scanner s=new Scanner(System.in);
    System.out.println("enter the day in a week");
    String s1=s.nextLine();
            switch(s1) {
            case "Monday" : System.out.println("Monday is upma");
            case "Tuesday" : System.out.println("Tuesday is Puri");
            case "Wednesday" : System.out.println("Wedensday is Parota");
            case "Thursday" : System.out.println("Thursday is Dosa");
            case "Friday" : System.out.println("Friday is pulav");
            case "Saturday" : System.out.println("Saturday is vada");
            case "Sunday" : System.out.println("Sunday is Yumm..Biryaniiii");

            }
}
}
```

```
<terminated> Program6 (3) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (24-Feb-2025, 8:50:31 pm – 8:50:35 p
enter the day in a week
Monday
Monday is upma
Tuesday is Puri
Wedensday is Parota
Thursday is Dosa
Friday is pulav
Saturday is vada
Sunday is Yumm..Biryaniiii
```

```java
package DMS;

import java.util.Scanner;

public class Program7 {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        System.out.println("please enter the character from 'a' to 'e'");
        char c=s.next().charAt(0);
        switch(c)
        {
        case 'a' : System.out.println("a for apple");
        case 'b' : System.out.println("b for ball");
        case 'c' : System.out.println("c for cat");
        case 'd' : System.out.println("d for dog");
        case 'e' : System.out.println("e for eagle");
                   break;
        default : System.out.println("please enter a valid character");
        }
```

```
<terminated> Program7 (1) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (24-Feb-2025, 8:56:45 pm – 8:56:4
please enter the character from 'a' to 'e'
c
c for cat
d for dog
e for eagle
```

```java
public class Program7 {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        System.out.println("please enter the character from 'a' to 'e'");
        char c=s.next().charAt(0);
        switch(c)
        {
        case 'a' : System.out.println("a for apple");
                   break;
        case 'b' : System.out.println("b for ball");
                   break;
        case 'c' : System.out.println("c for cat");
                   break;
        case 'd' : System.out.println("d for dog");
                   break;
        case 'e' : System.out.println("e for eagle");
                   break;
        default : System.out.println("please enter a valid character");
        }
```

```
<terminated> Program7 (1) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (24-Feb-2025, 8:57:54 pm – 8:5
please enter the character from 'a' to 'e'
c
c for cat
```

```java
    System.out.println("enter the day in a week");
    String s1=s.nextLine();
            switch(s1) {
            case "Monday" : System.out.println("Monday is upma");
                            break;
            case "Tuesday" : System.out.println("Tuesday is Puri");
                             break;
            case "Wednesday" : System.out.println("Wednesday is Parota");
                               break;
            case "Thursday" : System.out.println("Thursday is Dosa");
                              break;
            case "Friday" : System.out.println("Friday is pulav");
                            break;
            case "Saturday" : System.out.println("Saturday is vada");
                              break;
            case "Sunday" : System.out.println("Sunday is Yumm..Biryaniiii");
                            break;

            }
            System.out.println("byee byee switch");
```

```
<terminated> Program6 (3) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (24-Feb-2025, 8:52:06 pm – 8:52:09 pm
enter the day in a week
Wednesday
Wednesday is Parota
byee byee switch
```

```java
public class Program5 {
    public static void main(String[] args) {

        Scanner s=new Scanner(System.in);
        System.out.println("enter a number from 1 to 4");
        int a=s.nextInt();

    switch(a) {
    case 1 : System.out.println("i am one");
             break;
    case 2 :System.out.println("i am two");
              break;
    case 3 : System.out.println("i am three");
             break;
    case 4 : System.out.println("i am four");
              break;
    default : System.out.println("please enter the valid input");
```

```
<terminated> Program5 (3) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (24-Feb-
enter a number from 1 to 4
3
i am three
```

1)WAJP that takes an integer number as input from the user and check whether the number is positive , negative or zero

2)WAJP that takes an integer number as input from the user which represents age ranging from 0 to 100 and print the statements given if the age is between the following scenarios

if age is between 0-12: "I am a Child"
" 13-19: "Hey I am a Teenager"
" 20-64: "Hmm I am an Adult"
" 65 and above: "Hello I am Senior"

Note : if the user enter incorrect age then print please enter the valid age

3) WAJP that takes integer number from the user which represents the marks of students ranging from 0 to 100 and print the Grades if the marks are between the following Scenarios

if the marks are between 90 and above :"A"
" 70 to 89: "B"
" 50 to 69: "C"
" Below 50: "F"

4) WAJP that takes input from the user which represents a month number ranging from 1 to 12 and prints the corresponding month names using a switch statement.

Ex : 1 = "January" , 2="February".....etc

--> if the user enter 1 then you need to print all the month names from January to December

-->if the user enter 5 then you need to print all the months from may to December

Note : if the user enter number other than range then print "please enter the valid number"

5) WAJP that takes character input from the user which represents the 7 colours of rainbow "VIBGYOR"

Note : the user should enter the character which is present in VIBGYOR

if the user enters V then print Violet
I Indigo
B Blue
G Green
Y Yellow
O Orange
R Red

-->if the user enters the character which is not present in VIBGYOR then print "please enter the Character from the word VIBGYOR"

Loop Statements :

Case 1 :

Console

```
*
```

S.o.pln("*");

Case 2 :

```
*
*
*
*
```

S.o.pln("*");
S.o.pln("*");
S.o.pln("*");
S.o.pln("*");

S.o.pln("*") ⟶ 4 times

Def : it helps the programer to execute a instruction or set of instructions multiple times

Types of Loop Statements :

1) while
2) do while
3) for
4) for each / enhanced for

1) while : it is a keyword which is used as a Loop Statement

Syntax :

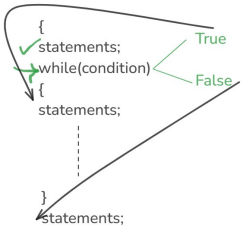
```
while(condition)
  statement;
```

⟶ if 1 statement to be repeated

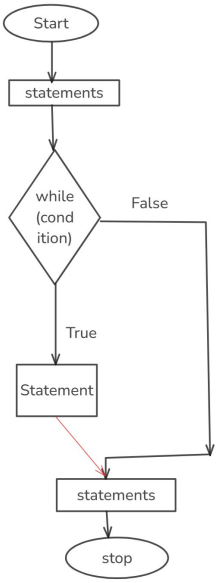
or

```
while(condition)
{
  statements;
  .
  .
}
```

⟶ if multiple statements to be repeated



Work Flow :



Ex :WAJP to print * for 5 times

```
int count=0;
while(count<5 )
{
S.o.pln("*");
count++;
}
```

count

count<5

0<5 True
1<5 True
2<5 True
3<5 True
4<5 True
5<5 false

count<=4

0<=4 True
1<=4 True
2<=4 True
3<=4 True
4<=4 True
5<=4 False

```
1
2
3
4
5
6
7
```

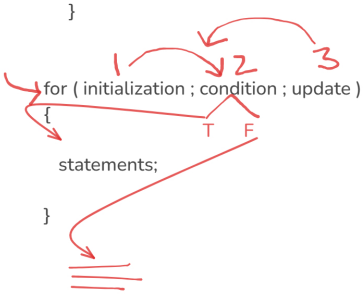
3 * 1 = 3
3 * 2 = 6
.
.
3 * 10 = 30

3) for : it is a keyword which is used as loop statement

Syntax :

start → for(initialization ; condition ; update)
statement; → end

```
for(initialization ; condition ; update )
{
statements;
}
```



Ex :

```
for( int i=1 ; i<=5 ;i++)
{
S.o.pln(i);
}
```

```
1
2
3
4
5
```

i	i<=5
1	1<=5 True
2	2<=5 True
3	3<=5 True
4	4<=5 True
5	5<=5 True
6	6<=5 False

Console

```
1
2
3
4
5
```

Nested Loop :

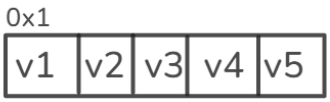
```
for(int j=1 ; j<=3 ; j++)
{
for ( int i =1 ; i<=3 ; i++)
{
S.o.p(i + " ");
}
S.o.pln();
}
```

```
1 2 3
1 2 3
1 2 3
```

I	j = 1	1<=3 True
		i) i = 1 — 1<=3 true
		ii) 2 — 2 <=3 true
		iii) 3 — 3<=3 true
		iv) 4 — 4<=3 false
II	2	2<=3 True
		i=1 — 1<=3 true
		2 — 2<=3 true
		3 — 3<=3 true
		4 — 4<=3 false
III	3	3<=3 true
		i=1 — 1<=3 true
		2 — 2<=3 true
		3 — 3<=3 true
		4 — 4<=3 false
IV	4	4<=3 false

```
1 2 3
1 2 3
1 2 3
```

Array : Array is a continuos block of memory which is used to store the multiple values

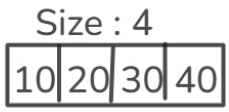


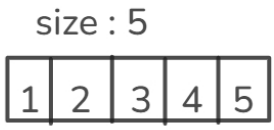
Characteristics of an Array :

1) Arrays are fixed in size

-->whenever we create / declare an array we need to mention size

-->we cannot change the size of an array once it is created

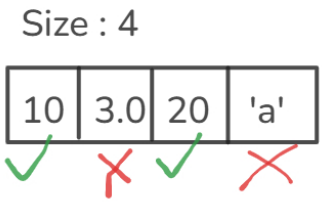




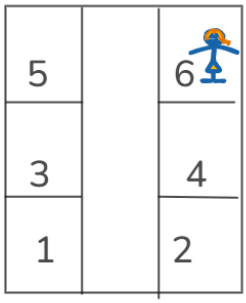
2)Array is Homogeneous Collection of Data

-->it will only accept the same type of Data

Note : we provide the datatype at the time of Array declaration



3)



Abc ,door no : 6 , Basavangudi,banglore, 500007

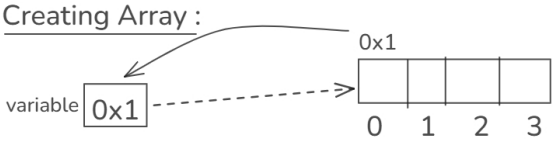
-->to store or fetch the values / elements in an Array we require

1) array reference

2)index

it is a +ve integer which always starts from 0 upto size-1

Creating Array :



to declare array reference variable :

syntax : datatype[ ] identifier;

datatype identifier[ ];

Ex : int[] a;

a=10;

a=null;

it doesnot mean empty
-->it is not refering to anyone

Ex : char[] ch;

ch='a';

ch=null;

Create an array using new :

syntax : new datatype[size]

ex : new int[4]

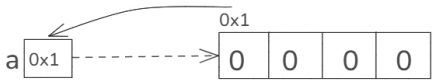
--> new creates the arry of given type for the given size and intialize with default values

--> it returns the reference of array

syntax :

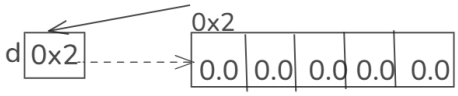
datatype[ ] variable=new datatype[size]

Ex : int[ ] a=new int[4];



S.o.pln(a);

Ex : double[] d=new double[5];



to create an array without using new :

Syntax :

datatype[ ] identifier = {value1,value 2.....valuen};

Ex : int[ ] a = {10,20,30,40}; → size is 4

-->the biggest advantage we can store the values directly

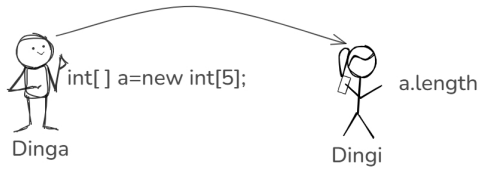
-->The size of an array is equal to the no.of values we pass

Ex : double[ ] d={1.2,3.5,6.5}; → size is 3

Ex : int[ ] a;

a={10,20,30,40};

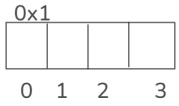
to obtain the length of an array :



Syntax : `array_reference.length`

to store and access the data in an array :

-->array reference
-->.index

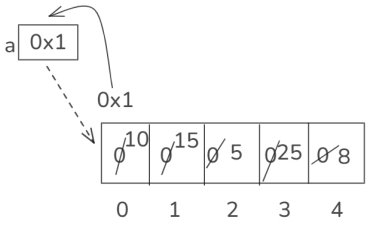


Syntax to store :

`array_reference[index] = value / variable / expression;`

Ex : int[ ] a = new int[5];

a[0]=10;
a[1]= 10+5; 15
int b=5;
a[2]=b; 5
a[3]=b*5; 25
int c=3;
a[4]=b+c; 8



to fetch / access the data :

syntax :

`array_reference[index]`

to fetch nth element :

array_reference[n-1];

1st ---> 0
2nd-->1

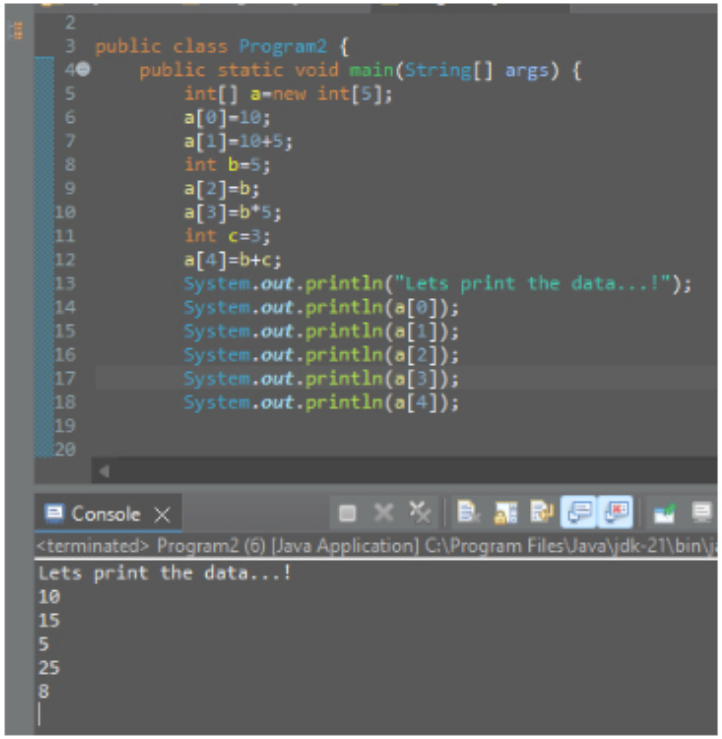
*) find the sum of 3rd element and 5th element

```
int sum=a[2]+a[4];
S.o.pln(sum);
```

ArrayIndexOutOfBoundsException :

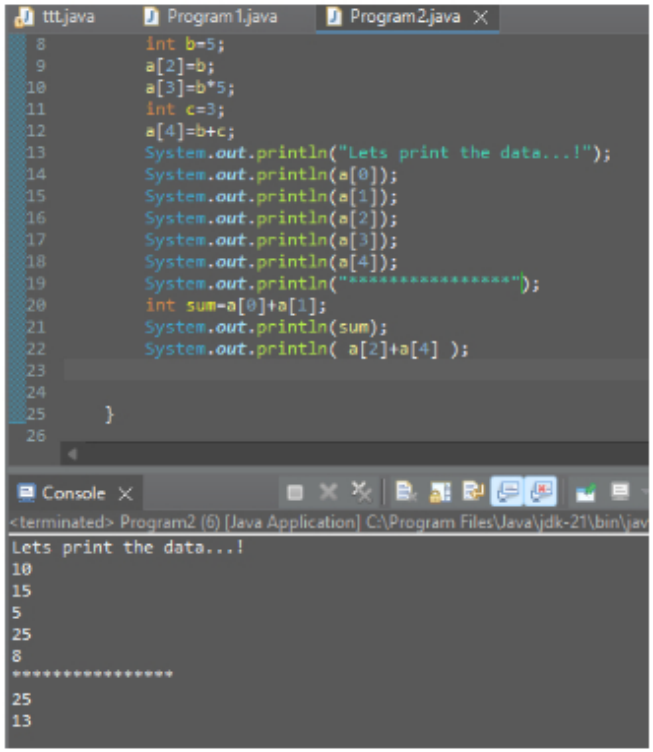
we get the AIOOBE when we try to access the element with the following index
-->if index < 0
-->if index = length of an array
-->if index > length of an array

Ex:
```
int[] a=new int[3];
a[0]=10;
a[1]=20;
a[2]=30;

S.o.pln(a[0]);//CTS  Run time
S.o.pln(a[-1]);//CTS  Run time exception
S.o.pln(a[3]);//CTS  run time exeption
S.o.pln(a[5]);//CTS run time exception
```



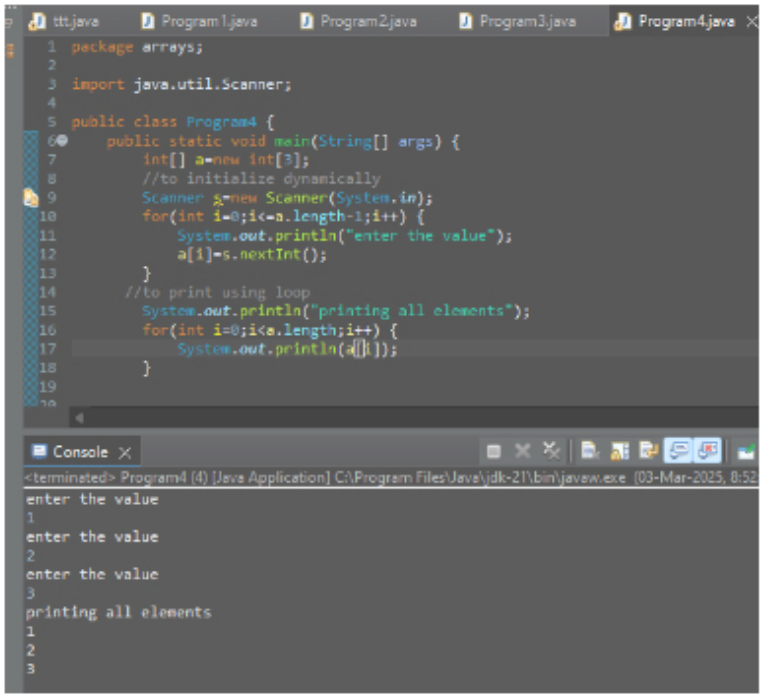
```java
public class Program2 {
    public static void main(String[] args) {
        int[] a=new int[5];
        a[0]=10;
        a[1]=10+5;
        int b=5;
        a[2]=b;
        a[3]=b*5;
        int c=3;
        a[4]=b+c;
        System.out.println("Lets print the data...!");
        System.out.println(a[0]);
        System.out.println(a[1]);
        System.out.println(a[2]);
        System.out.println(a[3]);
        System.out.println(a[4]);
```

```
Console
<terminated> Program2 (6) [Java Application] C:\Program Files\Java\jdk-21\bin\j
Lets print the data...!
10
15
5
25
8
```



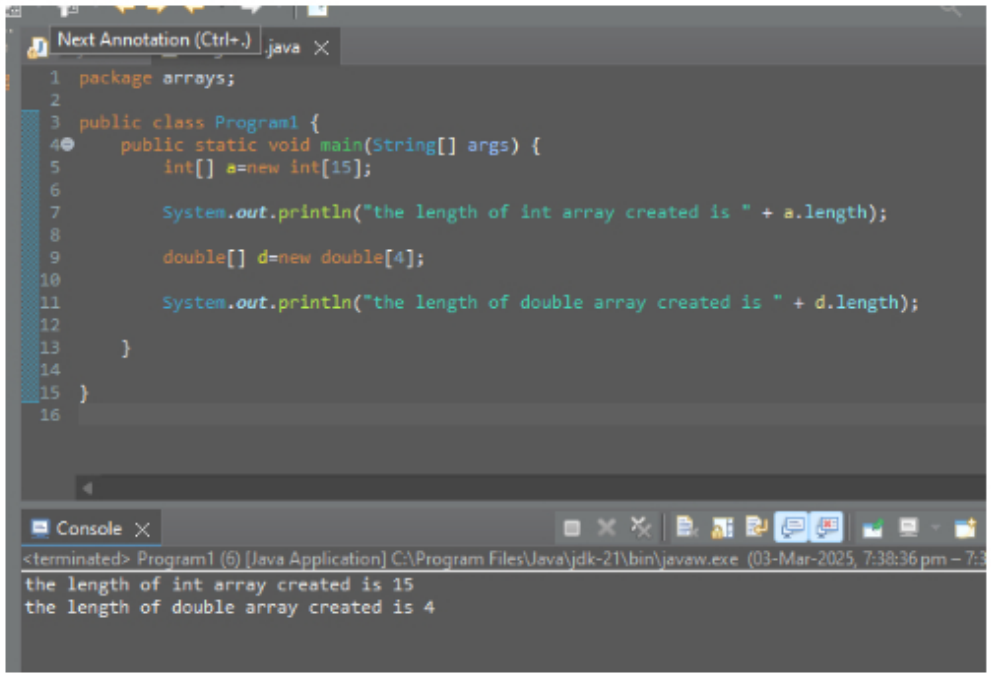
```java
        int b=5;
        a[2]=b;
        a[3]=b*5;
        int c=3;
        a[4]=b+c;
        System.out.println("Lets print the data...!");
        System.out.println(a[0]);
        System.out.println(a[1]);
        System.out.println(a[2]);
        System.out.println(a[3]);
        System.out.println(a[4]);
        System.out.println("****************");
        int sum=a[0]+a[1];
        System.out.println(sum);
        System.out.println( a[2]+a[4] );

    }
```

```
Console
<terminated> Program2 (6) [Java Application] C:\Program Files\Java\jdk-21\bin\jav
Lets print the data...!
10
15
5
25
8
****************
25
13
```



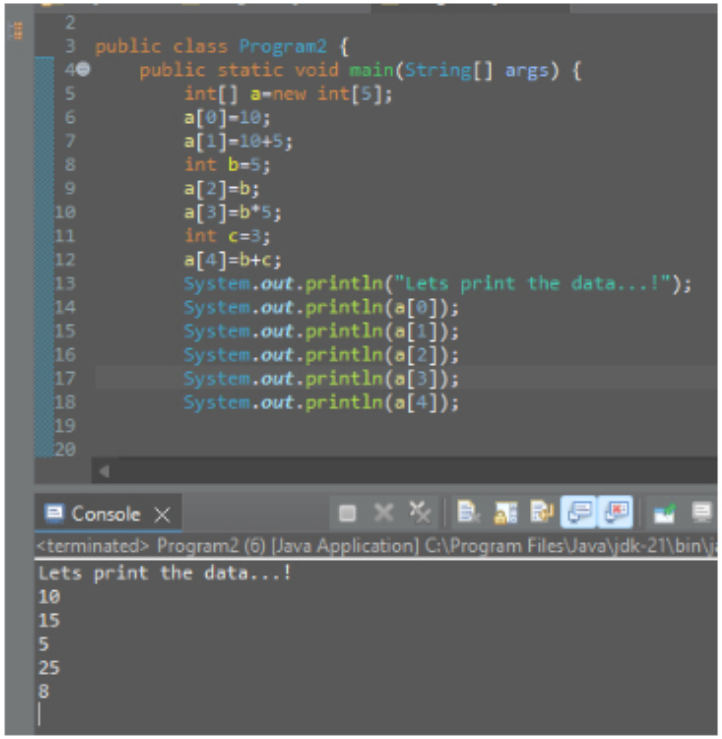
```java
package arrays;

import java.util.Scanner;

public class Program4 {
    public static void main(String[] args) {
        int[] a=new int[3];
        //to initialize dynamically
        Scanner s=new Scanner(System.in);
        for(int i=0;i<=a.length-1;i++) {
            System.out.println("enter the value");
            a[i]=s.nextInt();
        }
      //to print using loop
        System.out.println("printing all elements");
        for(int i=0;i<a.length;i++) {
            System.out.println(a[i]);
        }
```

```
Console
<terminated> Program4 (4) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (03-Mar-2025, 8:52
enter the value
1
enter the value
2
enter the value
3
printing all elements
1
2
3
```



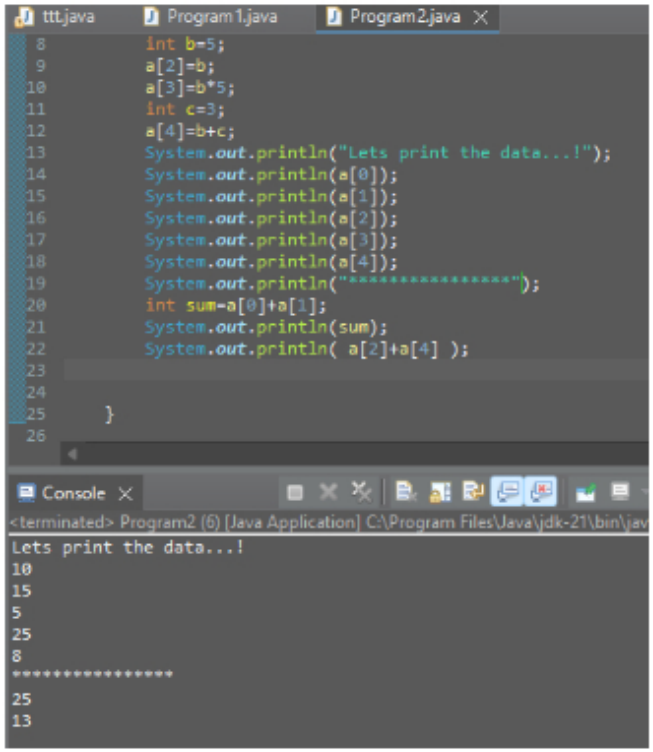
```java
package arrays;

public class Program1 {
    public static void main(String[] args) {
        int[] a=new int[15];

        System.out.println("the length of int array created is " + a.length);

        double[] d=new double[4];

        System.out.println("the length of double array created is " + d.length);

    }

}
```

```
Console
<terminated> Program1 (6) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (03-Mar-2025, 7:38:36 pm – 7:3
the length of int array created is 15
the length of double array created is 4
```



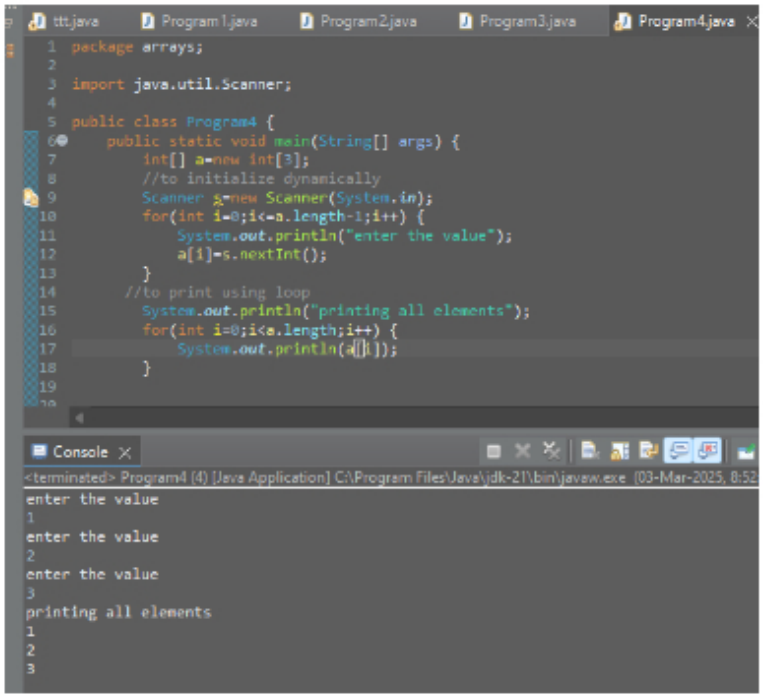
```java
public class Program2 {
    public static void main(String[] args) {
        int[] a=new int[5];
        a[0]=10;
        a[1]=10+5;
        int b=5;
        a[2]=b;
        a[3]=b*5;
        int c=3;
        a[4]=b+c;
        System.out.println("Lets print the data...!");
        System.out.println(a[0]);
        System.out.println(a[1]);
        System.out.println(a[2]);
        System.out.println(a[3]);
        System.out.println(a[4]);
```

```
Console
<terminated> Program2 (6) [Java Application] C:\Program Files\Java\jdk-21\bin\j
Lets print the data...!
10
15
5
25
8
```



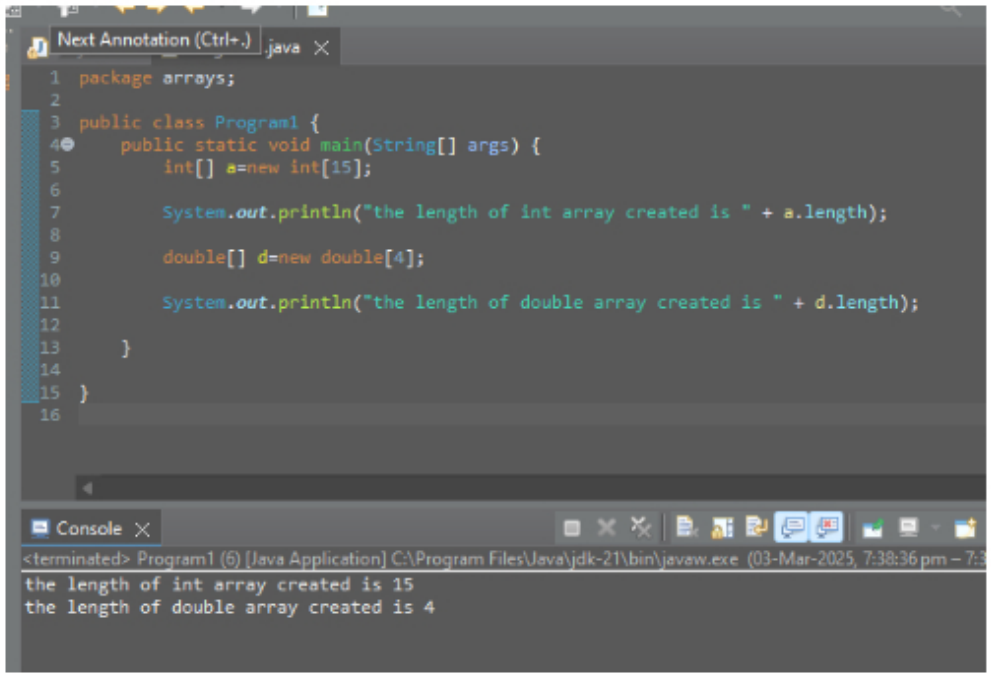
```java
        int b=5;
        a[2]=b;
        a[3]=b*5;
        int c=3;
        a[4]=b+c;
        System.out.println("Lets print the data...!");
        System.out.println(a[0]);
        System.out.println(a[1]);
        System.out.println(a[2]);
        System.out.println(a[3]);
        System.out.println(a[4]);
        System.out.println("****************");
        int sum=a[0]+a[1];
        System.out.println(sum);
        System.out.println( a[2]+a[4] );

    }
```

```
Console
<terminated> Program2 (6) [Java Application] C:\Program Files\Java\jdk-21\bin\jav
Lets print the data...!
10
15
5
25
8
****************
25
13
```



```java
package arrays;

import java.util.Scanner;

public class Program4 {
    public static void main(String[] args) {
        int[] a=new int[3];
        //to initialize dynamically
        Scanner s=new Scanner(System.in);
        for(int i=0;i<=a.length-1;i++) {
            System.out.println("enter the value");
            a[i]=s.nextInt();
        }
      //to print using loop
        System.out.println("printing all elements");
        for(int i=0;i<a.length;i++) {
            System.out.println(a[i]);
        }
```

```
Console
<terminated> Program4 (4) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (03-Mar-2025, 8:52
enter the value
1
enter the value
2
enter the value
3
printing all elements
1
2
3
```



```java
package arrays;

public class Program1 {
    public static void main(String[] args) {
        int[] a=new int[15];

        System.out.println("the length of int array created is " + a.length);

        double[] d=new double[4];

        System.out.println("the length of double array created is " + d.length);

    }

}
```

```
Console
<terminated> Program1 (6) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (03-Mar-2025, 7:38:36 pm – 7:3
the length of int array created is 15
the length of double array created is 4
```

Accessing the array elements using loop statements :

```
int[] a=new int[5];
        a[0]=10;
        a[1]=20;
        a[2]=30;
        a[3]=40;
        a[4]=50;
        System.out.println(a[0]);
        System.out.println(a[1]);
        System.out.println(a[2]);
        System.out.println(a[3]);
        System.out.println(a[4]);
```

```
                i<=a.length-1
for(int i=0 ; i<a.length ; i++)
{
    S.o.pln(a[i]);
}
```

0<5 T 3<5 T
1<5 T 4<5 T
2<5 T 5<5 F

Dynamic initalization of array :

```
int[] a=new int[5];
    a[0]=10;
    a[1]=20;
    a[2]=30;
    a[3]=40;
    a[4]=50;
```

```
int[] a=new int[5];
Scanner s=new Scanner(System.in);
    a[0]=s.nextInt();
    a[1]=s.nextInt();
    a[2]=s.nextInt();
    a[3]=s.nextInt();
    a[4]=s.nextInt();
```

```
int[] a=new int[5];
Scanner s=new Scanner(System.in);
 for(int i=0 ; i < a.length ; i++ )
 {
 a[i]=s.nextInt();
 }
```

Task :

1)Create 2 arrays of double type and char type with size 7
-->first store the elements
-->print the elements

-->Use the Scanner class for dynamic initialization
and print them

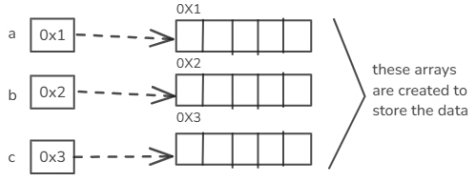
Multi Dimensional Array :

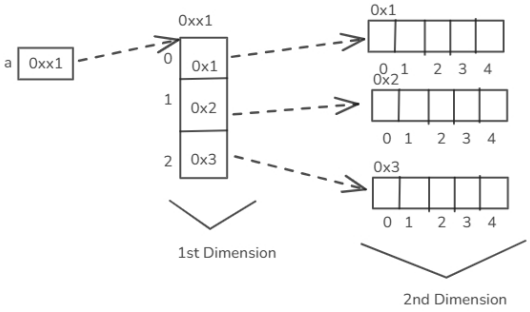
--> 2D Array is to store the data in the form of tables

rows and columns

1	2	3
4	5	6
7	8	9

3x3
RxC





to create Array Reference Variable :

Syntax :

```
Datatype[][]   identifer;
```

```
int[][] a;
a=10;
a=new int[3];
```

-->for 2D reference variable , we cannot store 1D reference

how to create array :

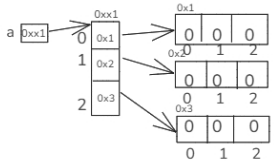
syntax :

```
new DataType[row][col];
```

(row: size, col: size; 1D, 2D)

Note:
Row size is mandatory
column size is optional
if we dont provide any column size array objects will not be created

Ex: int[][] a=new int[3][3];

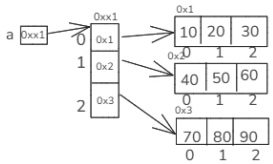



```
S.opln(a);

S.opln(a[0]);
S.opln(a[1]);
S.opln(a[2]);

S.opln(a.length);
S.o.pln(a[0].length);
S.opln(a[1].length);
S.opln(a[2].length);
```

to store the elements :

arrayreference[rowindex][columnindex]=value/variable/expresiion;

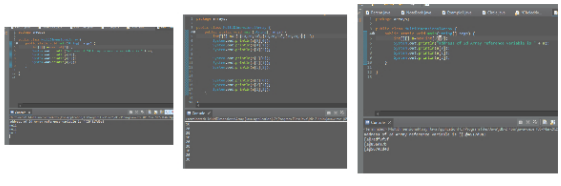
```
a[0][0]=10;
a[0][1]=20;
a[0][2]=30;
a[1][0]=40;
a[1][1]=50;
a[1][2]=60;
a[2][0]=70;
a[2][1]=80;
a[2][2]=90;
```



to fetch / access the elements :

```
arrayrefrence[rowindex][columnindex];
```

```
Sopln(a[0][0]);
Sopln(a[0][1]);
Sopln(a[1][2]);
Sopln(a[2][0]);
```



Program5.java | MultiDimensi... | Example.java | Test1.java

```java
package object_fundamentals;

class kavya{
    public static void main(String[] args) {
        int[] [] a= {{10,20},{20,30},{30,40}};
        for(int i=0;i<a.length;i++)
        {
            for(int j =0;j<a[i].length;j++)
            {
                System.out.print(a[i][j]+ " ");

            }
            System.out.println();
        }
    }
}
```

Console

<terminated> kavya (1) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (0

```
10 20
20 30
30 40
```

Object Oriented :

object :

it is a block of memory which can have multiple variables and methods
-->every object will have reference / address

why do we need object :

Real world

properties

ram : 4
rom :32
battery : 6000
brand: samsung
version : S23
color:black
price:40000

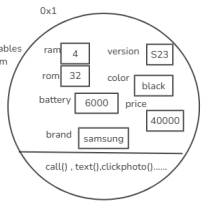
Actions
cal()
text()
gaming()
clickphoto()

Programing world

variables ✗
array ✗
objects ✓

-->with the help of object we can store different types of data (Heterogeneous data)
-->inside the object we can create multiple variables of different types

-->Properties in the form of variables
-->behaviours/ actions in the form of methods



Ex :

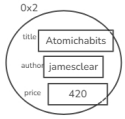
title
author
price

B1

wingsOfFire
abdul
300

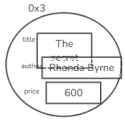
B2

Atomic Habits
james clear
420



B3

The Secret
Rhonda Byrne
600



to print price of B1
-->Sopln(0x1.price);

to print author of B1
-->Sopln(0x2.author);
to print the title of B1
--> Sopln(0x1.title);

to mofify the price of book by increasing of 100rs for B2

0x2.price=0x2.price+100;

Ex :

eid
name
salary

employee

E1

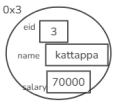
1
Romeo
15000

E2

2
Juliet
25000

E3

3
kattappa
70000



1)display all properties of E1 employee
Sopln(0x1.eid);
Sopln(0x1.name);
Sopln(0x1.salary);

2)display the E2 employee salary with deduction of 5000

0x2.sal=0x2.salary-5000;

Sopln(0x2.salary);

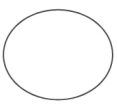
TASK :

Ex :

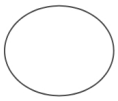
sid
sname
class

Student

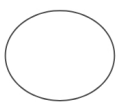
S1



S2



S3



Steps to create an object :

Step 1 : to create an object ,Blueprint of the object is neccesary

Bluprint exists :

create an object

Bluprint doesn't exist :

design the blueprint
create the object

Blueprint : it provide the specifications of an object

-->it tells what type of variables ,methods ,.....etc
no .of variables , names of the identifiers

-->we need a Blueprint to create an object

How to create a Blueprint :

->in java we can create Blueprint using class

class : it is a component of java
-->purpose of class
--to execute an application
--class behaves as an Blueprint of an object

-->we cannot create an object without the class

syntax of the class :

class ClassName → non-primitive datatype / user defined datatype
{

}

steps to create object :

1)create a class / use existing class
2)create an object for the class using
--new operator
--constructor

Syntax to create object :

-->name of constructor
similar to classname

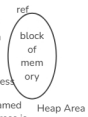
new :

--> new creates a block of memory in the heap area during the execution of java program

-->it is a run time memory allocation

-->new operator returns the reference/address

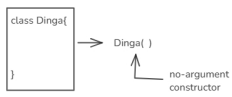
--> the memory which is created is not a named block, so it gives some address,the address is given by the new operator and also new operator returns the reference



Constructor :

-->it is a special member of the class whose name is similar to the ClassName

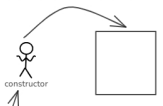
Ex :



-->whenever we create a class, Java Compiler will automatically create a constructor ,even if we dont create

Why we need it ?

```
class Sleeping{
//variables
//methods
static int i;
    int j;
}
```



it is used to load and initialize the non-static members of the class inside the object

Object Creation : Instantiation

2 steps to create an object

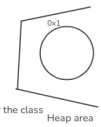
step 1:

Design a class /Blueprint

step 2 :

create an object

```
class Book
{
        new Book( );
}
```



-->the process of creating an object for the class is known as Instantiation

-->every time you say new, object will be created

-->object is also called as Instance of the class

```
Ex : class Example{
        public static void main(String[] args){
        Sopln("main starts");
        Sopln(new Example());
}
}
```

```
Ex : class Book{
        public static void main(String[] args){
       Sopln("main starts");
       Sopln(new Book());
       Sopln(new Book());
       Sopln(new Book());
        Sopln("main ends");
}
}
```

```
Ex : class Example2{
       public static void main(String[] args){
       sopln(new Example2());
       }
       }
```

Note : we can create object for a class inside the same class or in a different class.

-->for a class, we can create any no.of objects

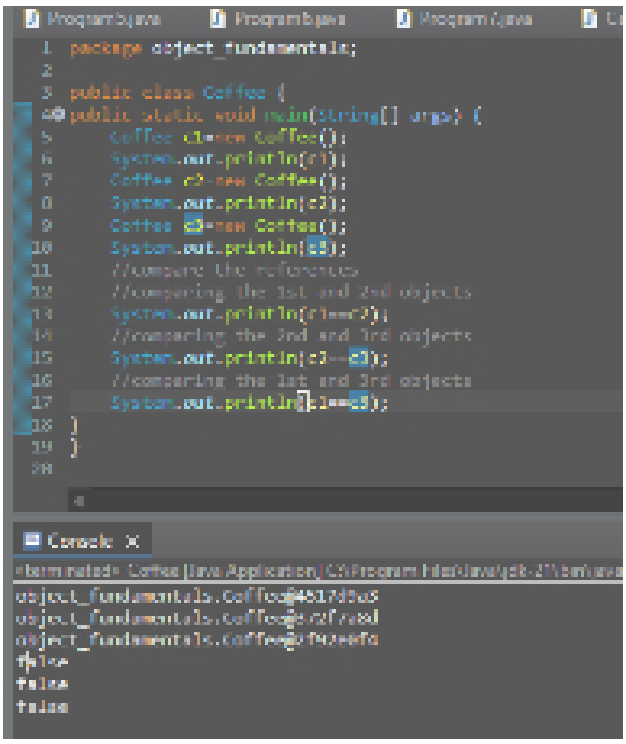
Non-Primitive variable / reference variable :

-->if we want to use the variable/method inside the object , reference is mandatory

-->new operator will return the reference
-->we need to store the reference/ address , to store the address, we need a memory ,we need to create a variable to store the address i.e., Non-primitive variable / refrence variable

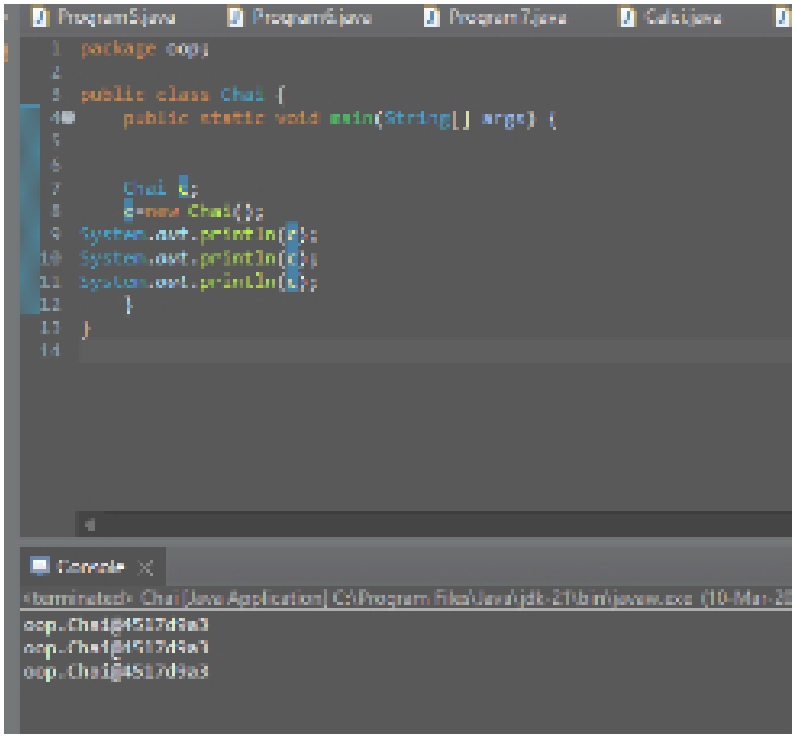
--> here we can store the reference of an object

Def : The variable which is used to store the reference of an object is called as Reference variable / Non-Primitive variable



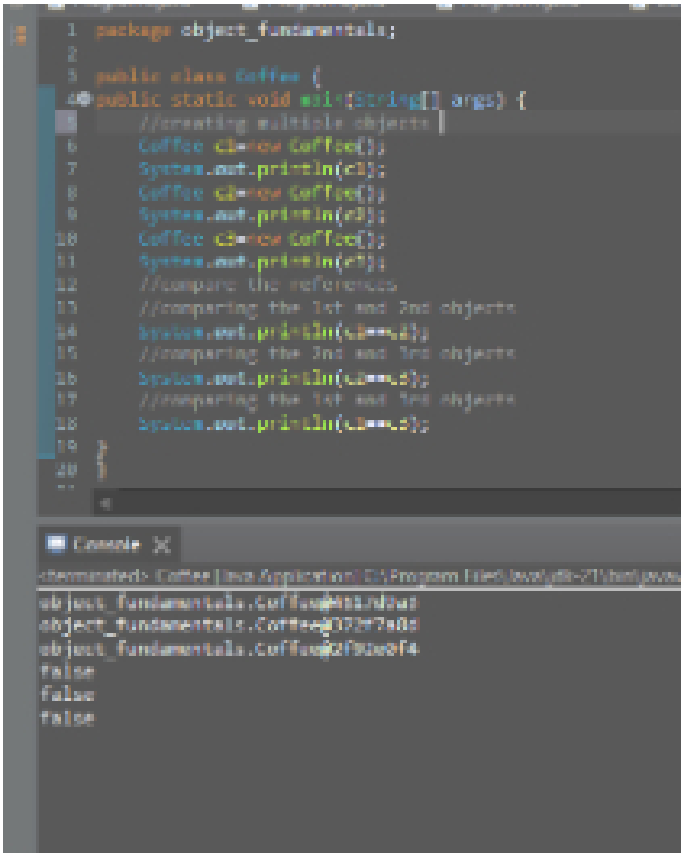
```
package object_fundamentals;

public class Coffee {
public static void main(String[] args) {
    Coffee c1=new Coffee();
    System.out.println(c1);
    Coffee c2=new Coffee();
    System.out.println(c2);
    Coffee c3=new Coffee();
    System.out.println(c3);
    //compare the references
    //comparing the 1st and 2nd objects
    System.out.println(c1==c2);
    //comparing the 2nd and 3rd objects
    System.out.println(c2==c3);
    //comparing the 1st and 3rd objects
    System.out.println(c1==c3);
}
}
```

```
object_fundamentals.Coffee@4517d9a3
object_fundamentals.Coffee@372f7a8d
object_fundamentals.Coffee@2f92e0f4
false
false
false
```



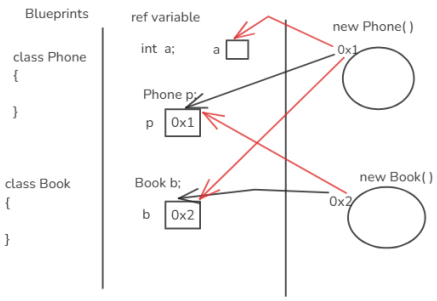
```
package oop;

public class Chai {
    public static void main(String[] args) {


    Chai c;
    c=new Chai();
System.out.println(c);
System.out.println(c);
System.out.println(c);
    }
}
```

```
oop.Chai@4517d9a3
oop.Chai@4517d9a3
oop.Chai@4517d9a3
```



```
package object_fundamentals;

public class Coffee {
public static void main(String[] args) {
    //creating multiple objects
    Coffee c1=new Coffee();
    System.out.println(c1);
    Coffee c2=new Coffee();
    System.out.println(c2);
    Coffee c3=new Coffee();
    System.out.println(c3);
    //compare the references
    //comparing the 1st and 2nd objects
    System.out.println(c1==c2);
    //comparing the 2nd and 3rd objects
    System.out.println(c2==c3);
    //comparing the 1st and 3rd objects
    System.out.println(c1==c3);
}
}
```

```
object_fundamentals.Coffee@4517d9a3
object_fundamentals.Coffee@372f7a8d
object_fundamentals.Coffee@2f92e0f4
false
false
false
```



How do we create a reference variable ??

we can create reference variable with the help of non-primitive datatype

What can i store inside the reference variable??

we can store the refernce or null(default value for non-primitive datatype)

Ex: class Book
{
}

TYPE 1:
Book b1;
b1=new Book( );
Sopln(b1);//0x1
Sopln(b1);//0x1

Type 2 :
class Book
{
}

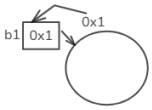
Book b=new Book( );
Sopln(b);

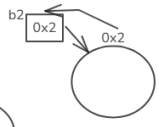
b 0x1

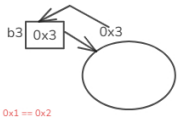
creating multiple objects for the class ?

class Book
{
}

Book b1=new Book( );
Book b2=new Book( );
Book b3=new Book( );







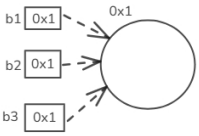
0x1 == 0x2
Sopln(b1==b2); //false
Sopln(b1==b3);//flase
0x1 == 0x3
Sopln(b2==b3);//false
0x2 == 0x3

== is used to compare the reference of 2 objects

Object refered by multiple variables :

class Book
{
}

Book b1=new Book( );
Book b2=b1;
0x1
Book b3=b2;
0x1



Sopln(b1==b2); // true
0x1==0x1

Sopln(b1==b3); // true
0x1==0x1

Sopln(b2==b3) ;// true
0x1==0x1

-->we can create multiple variables refering to one object

non-static variable :

A variable which is declared in the class block without prefixed with the static keyword / static modifier is known as non-static variable

Syntax :

class ClassName
{
datatype identifier;
}

class block

non-static variable

Ex 1 :

class Book
{
double price;
}

Ex 2 :

class Phone
{
String brand;
int ram;
}

non-static non-primitive variable

non-static primitive variable

Ex 3 :

class Demo
{
{
int i;
}
}

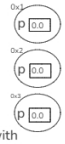
not a non-static variable

Characteristics of non-static variable :

1)memory is allocated inside the object
2)we cannot use a non-static variable without creating an object (instance) of the class
3)non-static variable will be allocated once in every object created for the class.

class Book
{
double p;
}

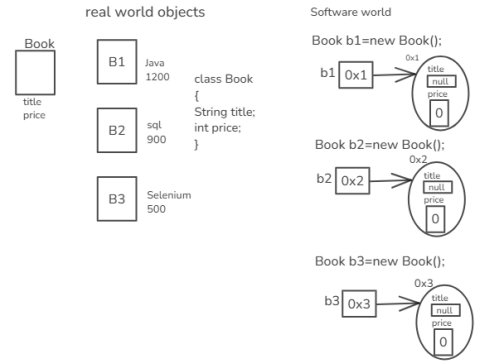
new Book( );
new Book( );
new Book( );



4)non-static variables are initialized with default values

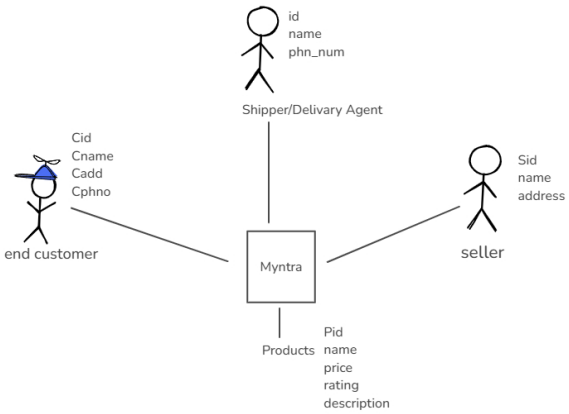
5)we can use a non-static variable with the help of object reference

working with non-static variables :



Ex : class Book
{
String title;
int price;
}

class BookDriver
{
public static void main(String[ ] args)
{
Book b1=new Book();
Book b2=new Book();
Book b3=new Book();
//to assign the data
b1.title="Java";
b1.price= 1200;
b2.title="Sql";
b2.price=900;
b3.title="Selenium";
b3.price=500;
//to print the properties
Sopln(b1.title);
Sopln(b1.price);
Sopln(b2.title);
Sopln(b2.price);
Sopln(b3.title);
Sopln(b3.price);
}
}



Ex : class Products{
// create 5 non-static variables for 5 properties
//Pid,name,price,rating,desc
int Pid;
String name;
double price;
int rating;
String desc;
}

class ProductsDriver{
public static void main(String[] args){
//step 1 : to create an object
Products p1=new Products();
Products p2=new Products();
//step 2 : intialize for 1st object
p1.Pid=1;
p1.name="Shirt";
p1.price= 1500;
p1.rating =4;
p1.desc="Men Regular Fit Solid Button Down Collar Formal Shirt ";
// for 2nd object
p2.Pid=143;
p2.name="Sareee";
p2.price=2000;
p2.rating=5;
p2.desc="Women Georgette Saree";
//step 3 : to print the data
Sopln(p1.Pid);
Sopln(p1.name);
Sopln(p1.price);
Sopln(p1.rating);
Sopln(p1.desc);
//for 2nd object
Sopln(p2.Pid);
Sopln(p2.name);
Sopln(p2.price);
Sopln(p2.rating);
Sopln(p2.desc);
}
}

Constructor :

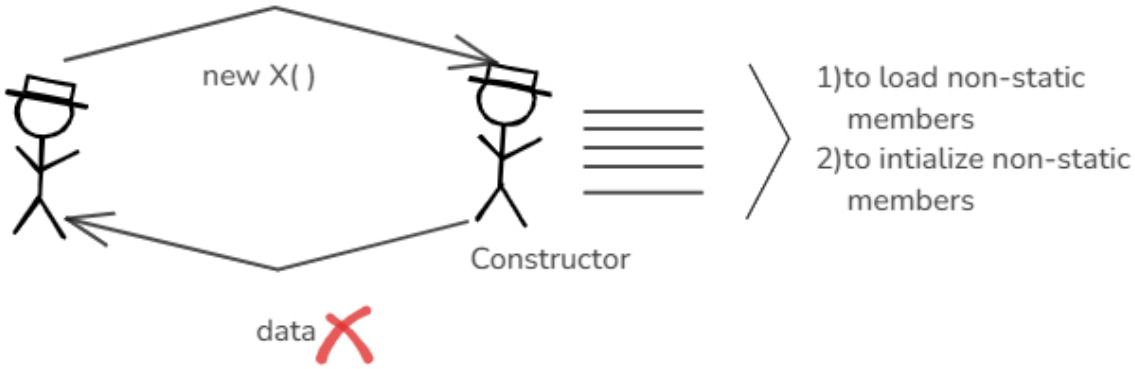
it is a special non-static member of a class which is used to load and initialize the non-static members

Characteristics :

1)it is a non-static Initializer

2)the name of the constructor is always similar to ClassName(including the cases)

3)Constructor doesnot return anything after execution , hence Constructor has no return type



to create a Constructor :

As a Programer we can define and create a constructor is known as User-Defined Constructor

Syntax :

[access_modifer] ClassName([datatype i ,.....])
{
Constructor Body
}

Ex :
```
class Thursday{
   Thursday( )
  {
    Sopln("Helloooo");
   }
 }
```

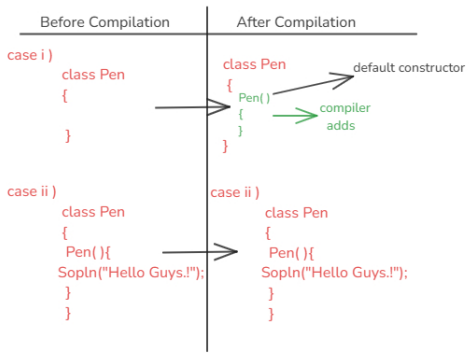
Constructor
no-arg Constructor
user-defined constructor

```
class ThursdayDriver{
public static void main(String[]  args){
Thursday t1=new Thursday( );
Sopln("**************");
Thursday t2=new Thursday( );

}
}
```

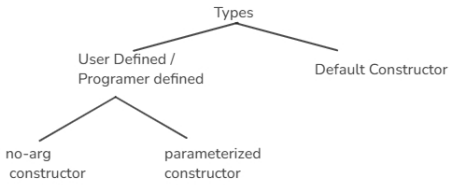
Console

```
Helloooo
**************
Helloooo
```

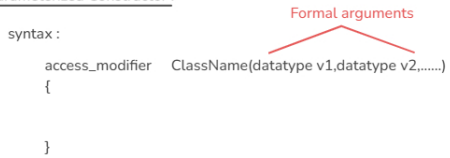
Note : if a programer fails to create a constructor
Compiler will add a no-arg constructor , it is
known as Default Constructor.



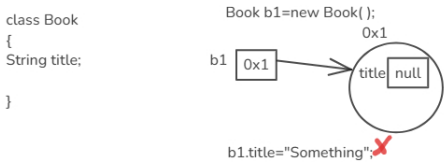
Types of Constructor :



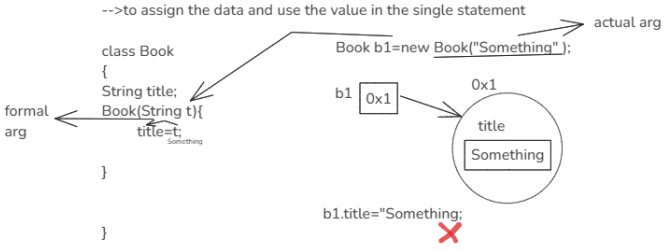
Parameterized Constructor :



why do we need a Parameterized Constructor :

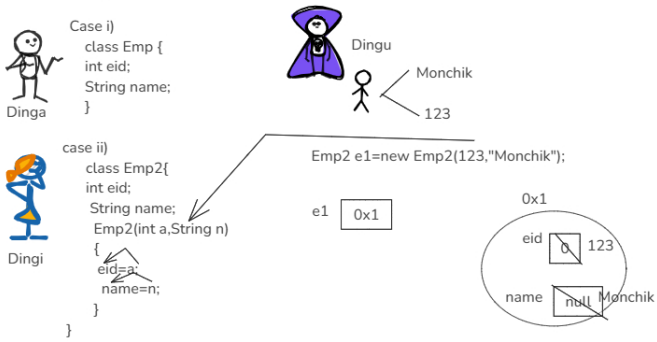


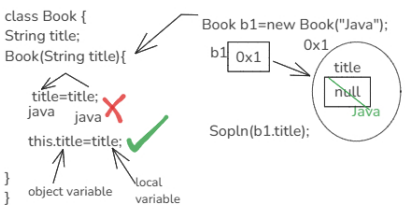
-->to assign the data and use the value in the single statement



Advantage of Parameterized Constructor :

1)It is used to intialize the object variables at the time of
Object Creation



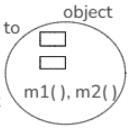


-->this will be pointing towards current object

Non-static method :

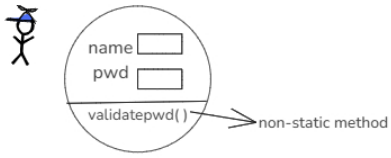
-->The methods which belong to the object is called as non- static method

--.the method which is present in the object is called as non-static method



Why do we need a Non-static method ?

-->non-static method represents action of an object



--> The non-static methods are generally used to access the data of that particular object

-->The main purpose of the non-static method is to access the properties of the object that could be reading,validating, modifying....

to Create a non-static method :

method created without static modifier as prefix

Syntax :

```
class Demo
{
void test( ){
}
}
```

Characteristic of non-static method :

1)Non-static method is allocated inside the object
2)we cannot call/use a non-static method without creating an object
3) we can call/use a non-static method with the help of Object Reference
4)A non-static method will be allocated inside every object created for the class

```
class Student{
void study( ){
}
}
```

```
new Student( );
new Student( );
```

5) A block which belongs to non-static method is known as Non-static context

```
class Demo{
void sleep( )
{
}
}
```

non-static context

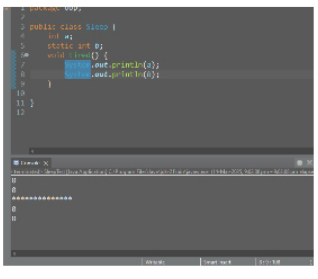
-->inside the non-static context we can use the non-static variables
-->inside the non-static context we can use the static variable
6)inside the non-static context we can use both the static and non-static variable directly without any reference

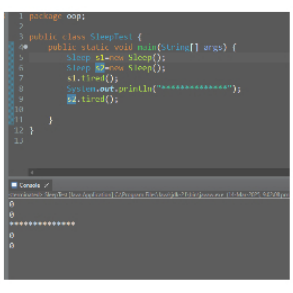
```
class Sleep {
int a;
static int b;
void tired( )
{
Sopln(a);
Sopln(b);
}
}
```

non-static context

non-static variable

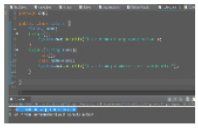
static-variable

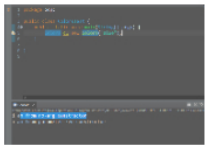




this( ): it is used to call the Constructor of same class.

Ex : class Demo {
Demo( ){
}
Demo(int i){
this( );
}
}





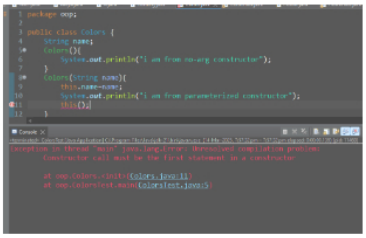
String name;
String color;
int price;

```
Flower(){
        System.out.println(name);
        }
Flower(String name){
        this.name=name;
        System.out.println(name);
}
Flower(String name,String color){
        this(name);
        this.color=color;
        System.out.println(color);
}
```

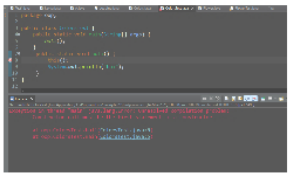
Flower f=new Flower("Lily","white");

Rules to use this( );

1) this( ) should be the first statement in the constructor body



2) this ( ) can be used only inside the Constructor body



3) Recursive call to this() is not allowed

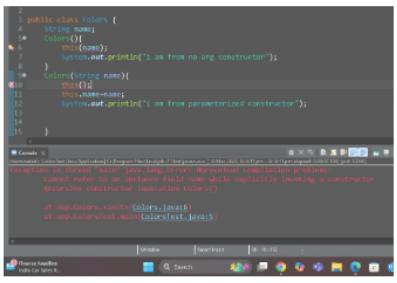
```
class Demo{
  Demo( ){
  this(i);
  }
  Demo(int i){
  this();
  }
  }
```

Demo d=new Demo(i);

Recursive call

Ex : class Demo{
Demo( ){
this( );
}
}

Note : we can use only (n-1) of constructors for this( )



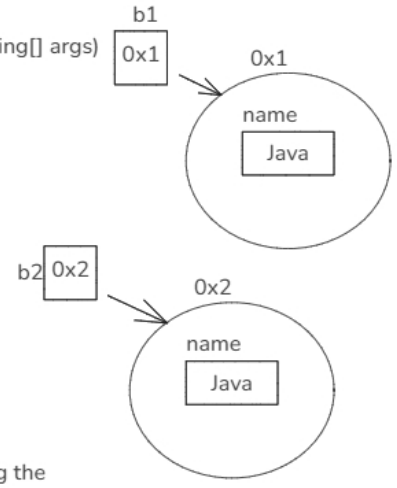
Non-static Initializer :

-->anything we do in the begining is called as Initializer

-->A single line instruction or multiline instruction which is used to execute during the Loading Process of an object

Single line non-static Initializer	Multi line non-static Initializer

Syntax :
class Name{ 2
datatype variable=value/exp;
1
}
Ex : class Book{
String name="Java";
}
	Syntax : class Name
	{
	{ stmts } Anonymous block/
	non-static block/
	Instance Initializer Block(IIB);
	}
	Ex : class BookDriver{
	public static void main(String[] args)
	{
	Book b1=new Book();
	Sopln(b1.name);
	Book b2=new Book();
	Sopln(b2.name);
	}



Characteristics of non-static Initializer :

1)non-static initializers get executed automatically during the object creation (Loading process of an object).
2)non-static initializer gets executed once for every object created

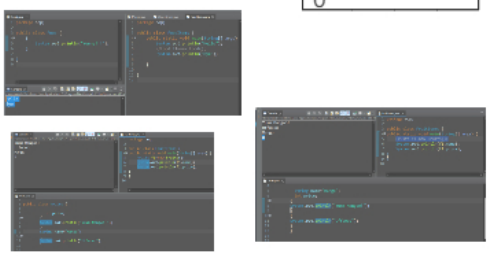
```
Ex : class Food{
      //IIB
     {
      Sopln("Yummyyy...!!");
     }
 }

 class FoodItems{
  public static void main(String[] args)
 {
 Food f1=new Food( );
 Sopln("Hello");
 }
 }
```

3)if class has more than 1 non-static initlaizer they get executed from top to bottom order

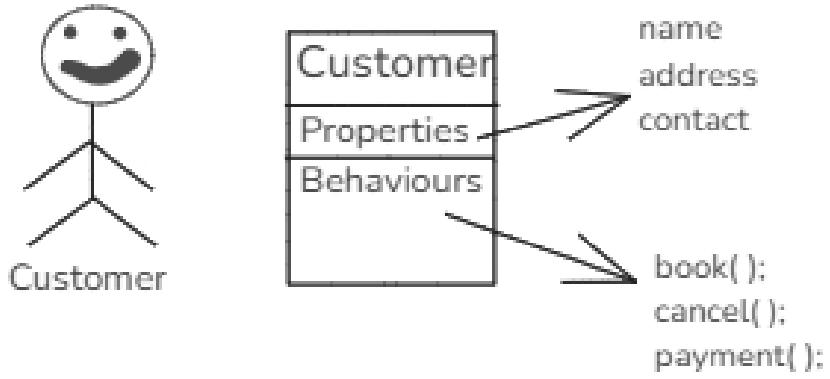
```
 class Fruits{
  String name="Mango";
  int price;
 {
 Sopln('Yumm Mangoo!");
 }
 {
 Sopln("Uffoooo");
 }
 }
```

```
Fruits f1=new Fruits();
Sopln(f1.name);
Sopln(f1.price);
```

```
Yumm Mangoo!
uffoooo
Mango
0
```



--> Object oriented is Bottom-up design



Step 1 : we have to analyse the real world object

--> identify properties

-->identify behaviours

Step 2 : Design a Blueprint

Step 3 : Instantiate the class

-->create object for the class in same class or different class

Design Principles :

There are 4 Design Principles of Object Oriented Programming

1.Encapsulation

2.Inheritance

3.Polymorphism

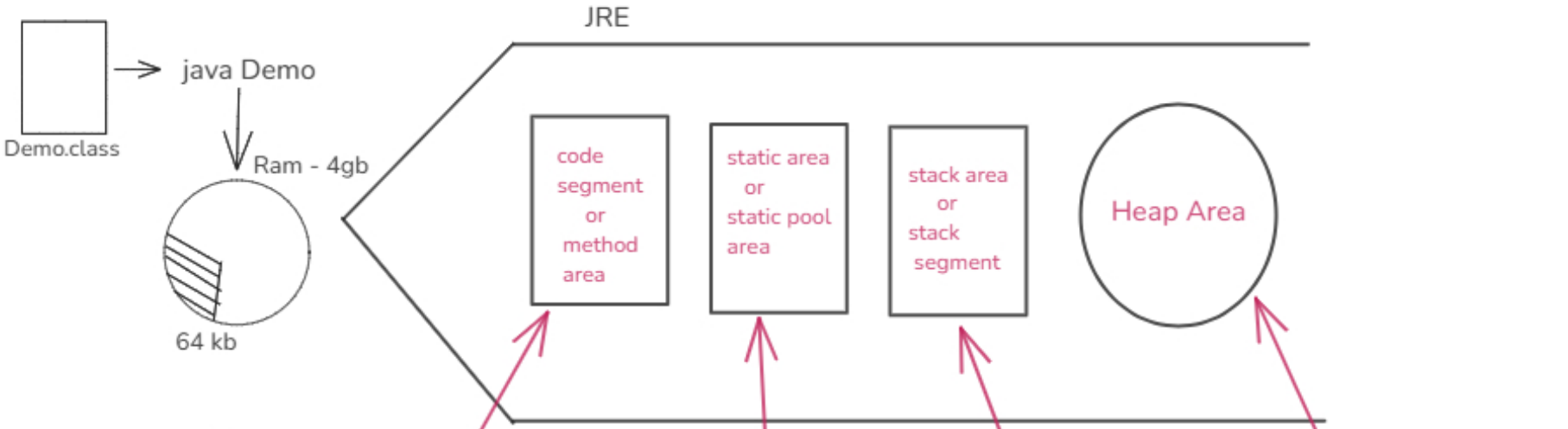
4.Abstraction

static :

-->it is a non-access modifier
-->it is used to create static members / class members

-->static variables
-->static methods
-->static blocks(static initializer)
-->static nested class

```
class Demo
{
//static varibles
//static methods
//static blocks
//static nested class
}
```




```
class
Hello{
---------
---------
}
```

all the instructions of the class will be stored here

static members will be here

used for execution of a method
-->local variables for the method will be created here

Objects that we create for a class , block of memory will be created inside this

static method :

Any method which is prefixed with static keyword

syntax :

```
class Demo
{
[modifier] static returntype name([Parameters..])
{

}
}
```

Ex :

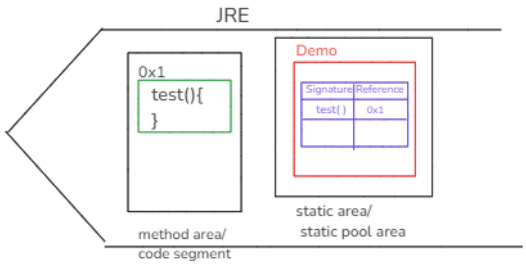
```
class Demo
{
 static void test( )
 {
  Sopln("Hello static method.!");
 }
}
```

static context

static method

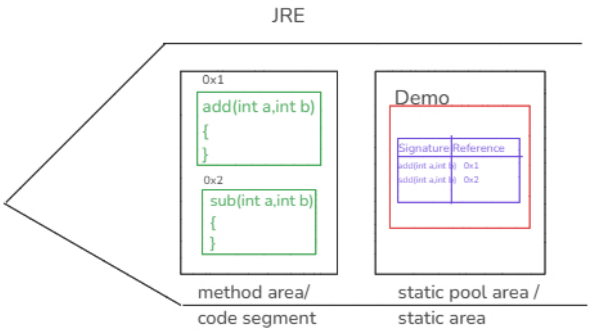
Ex:

```
class Demo
{
static void test( )
{
Sopln("Hello");
}
}
```



Ex :

```
class Demo
{
static int add(int a,int b)
{
return a+b;
}
static int sub(int a,int b)
{
return a-b;
}
}
```



Characteristics of a static method :

1)static method , memory will be allocated in the class Block
2)We can use a use / invoke a static method without creating an object for the class
3)We can call / invoke a static method within the same class directly with the help of method signature

```
Ex :  class Hello{
        public static void demo( ){
        Sopln("hai demo( )");
        }
        public static void main(String[] args){
          demo( );
          Hello.demo();
        }
      }
```

we can call it directly or with the help of class name

4)A static method of a class can be used in a different class with the help of classname as reference as well as with the help

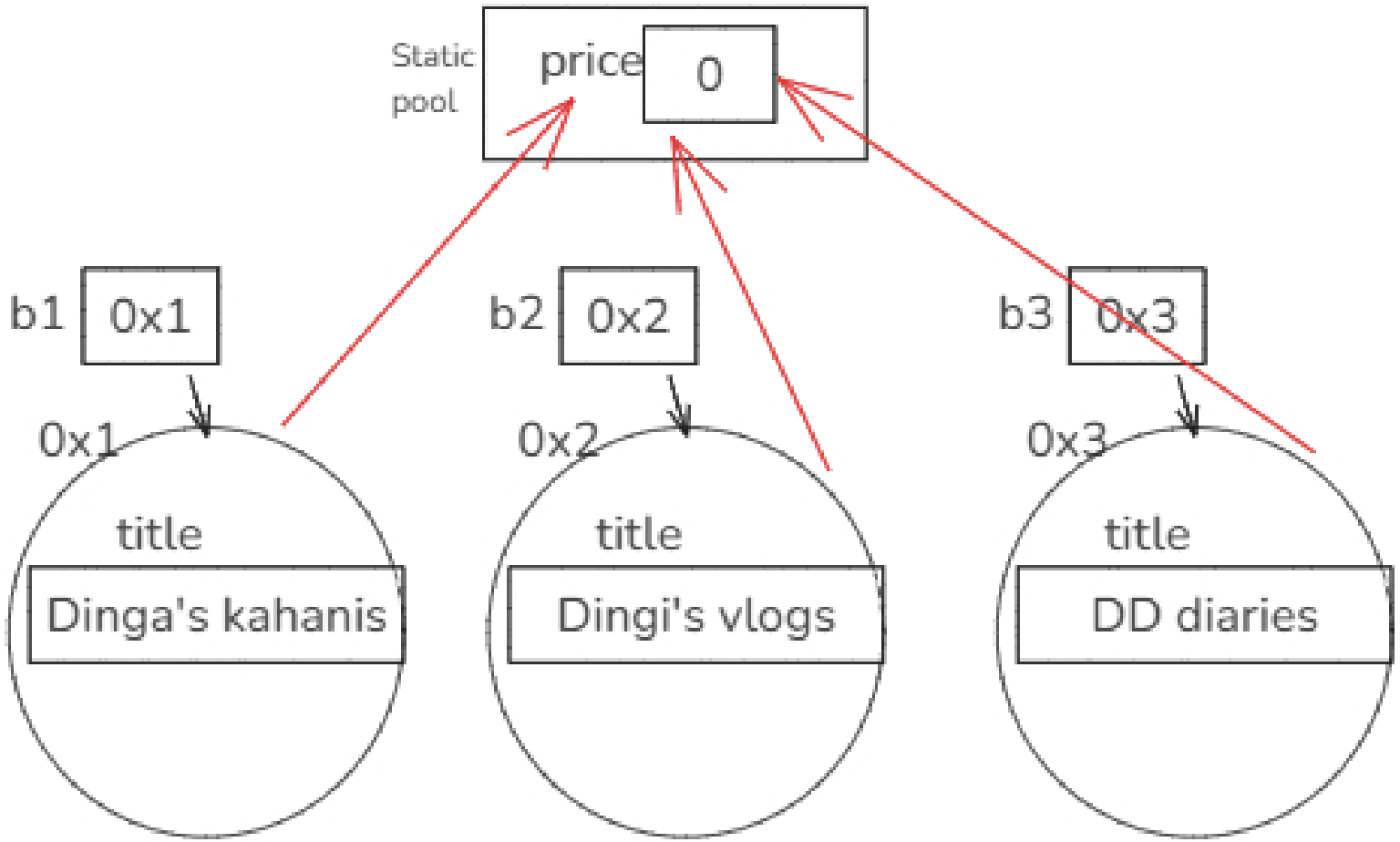
```
Ex : class Hello{
      p s v demo( )
      {
        Sopln("hai");
      }
     }
```

```
class HelloTest{
 p s v m (S[] a)
{
Hello.demo();
//using object reference
Hello h=new Hello();
h.demo();
}
}
```

5)the memory for the static variable is allocated only once
irrespective of how many objects created for the class

```
class Book{
static int price;
String title;
Book(String title)
{
this.title=title;
}
}
```

```
Book b1=new Book("Dinga's kahanis");
Book b2=new Book("Dingi's vlogs");
Book b3=new Book("DD diaries");
```



static Intializer :

-->There are 2 types
1)Single line static Initializer

syntax :

```
static datatype var=value / expresion;
```

2)Multi line static Initializer

syntax :

```
class Demo {
static {
----
----
}
}
```

multi line static initializer/
static block/
Static Intializer Block

static anonymous block

Characteristics of a static Intializer / static block :

1)static intializers get executed during Loading process of a class.

--> the entire class willl be brought from the hardisk to the JRE inside the ram is called as Loading process of a class

2)static intializer is executed only once

3)if a class has multiple static initialziers they gets executed from top to bottom order

Ex :

```
class Example
{
static
{
Sopln("from SIB 1 ");
}
 public static void main(String[] args)
{
Sopln("from main");
}
}
```

```
from SIB 1
from main
```

Ex :

```
public class Example1 {
    static {
        System.out.println("from SIB 1");

    }
    public static void main(String[] args) {
        System.out.println("from main");
    }
    static {
        System.out.println("from SIB 2");
    }

}
```

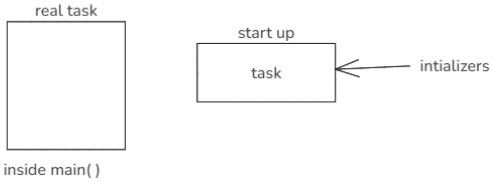
```
from SIB 1
from SIB 2
from main
```

Ex :

```
public class Example2 {
    static int a=test();
    public static int test() {
        System.out.println("from test");
        return 10;
    }
    public static void main(String[] args) {
        System.out.println("from main");
        System.out.println(a);
    }

}
```

```
from test
from main
10
```

Why do we need a static intializer :

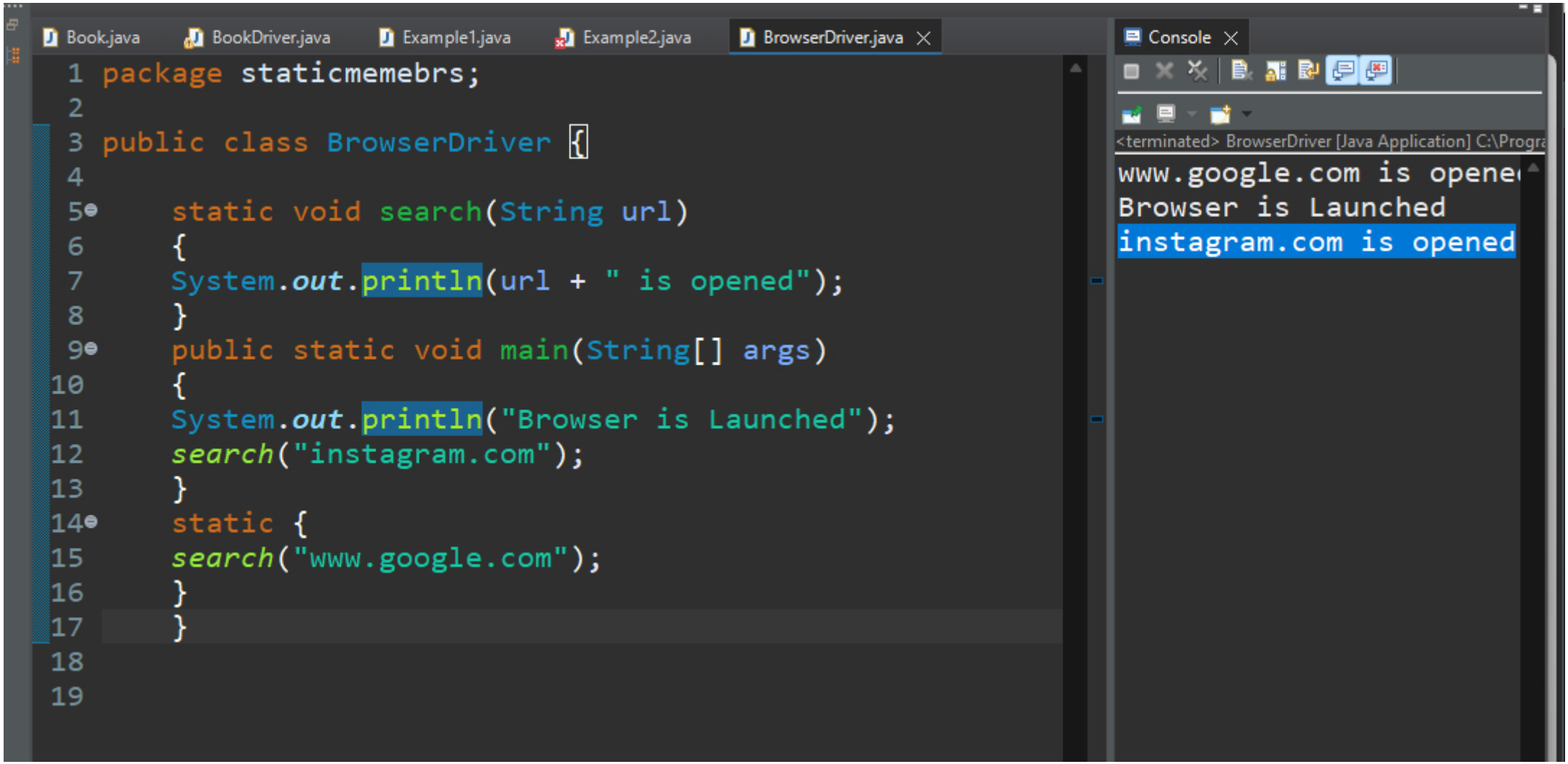


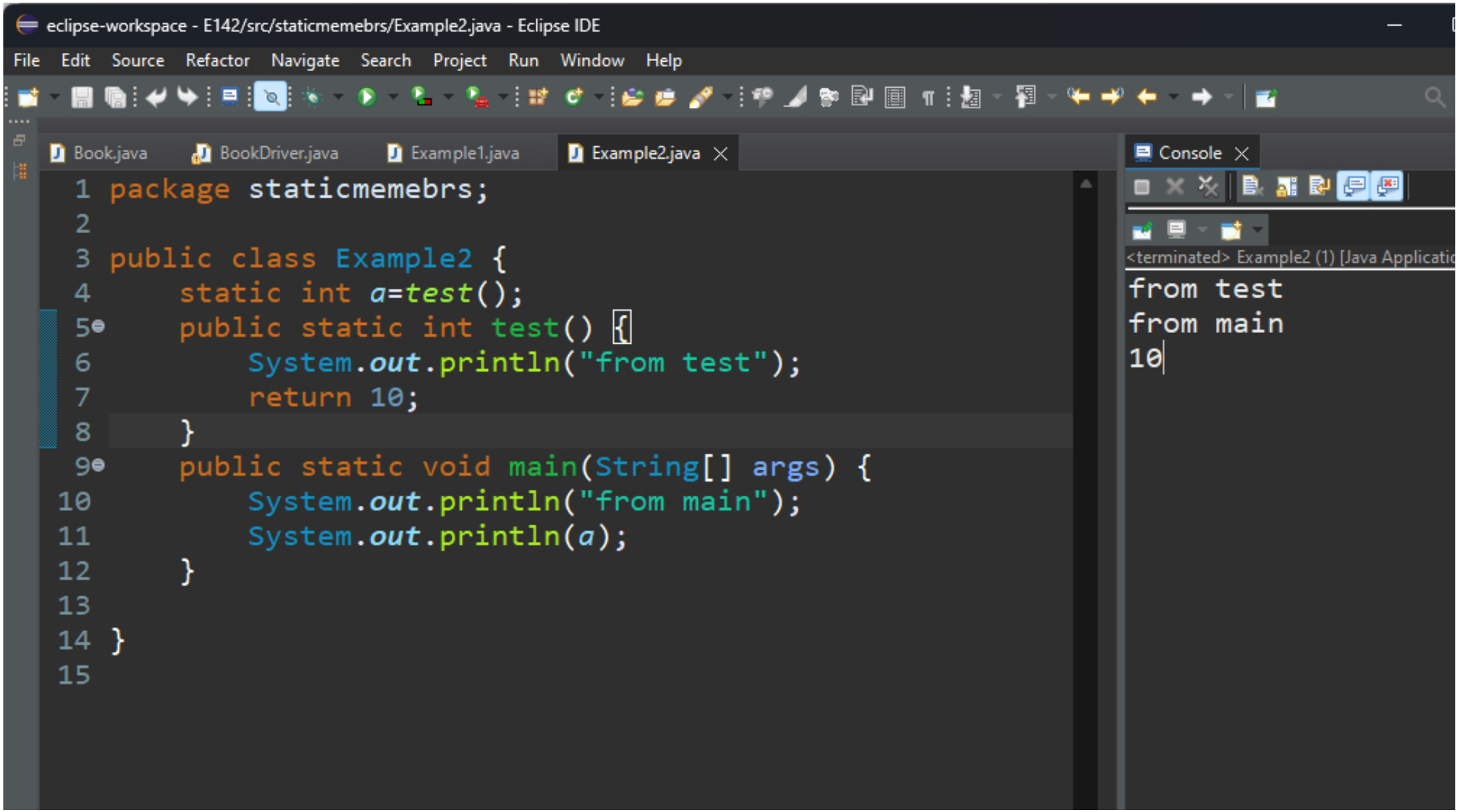
Ex :

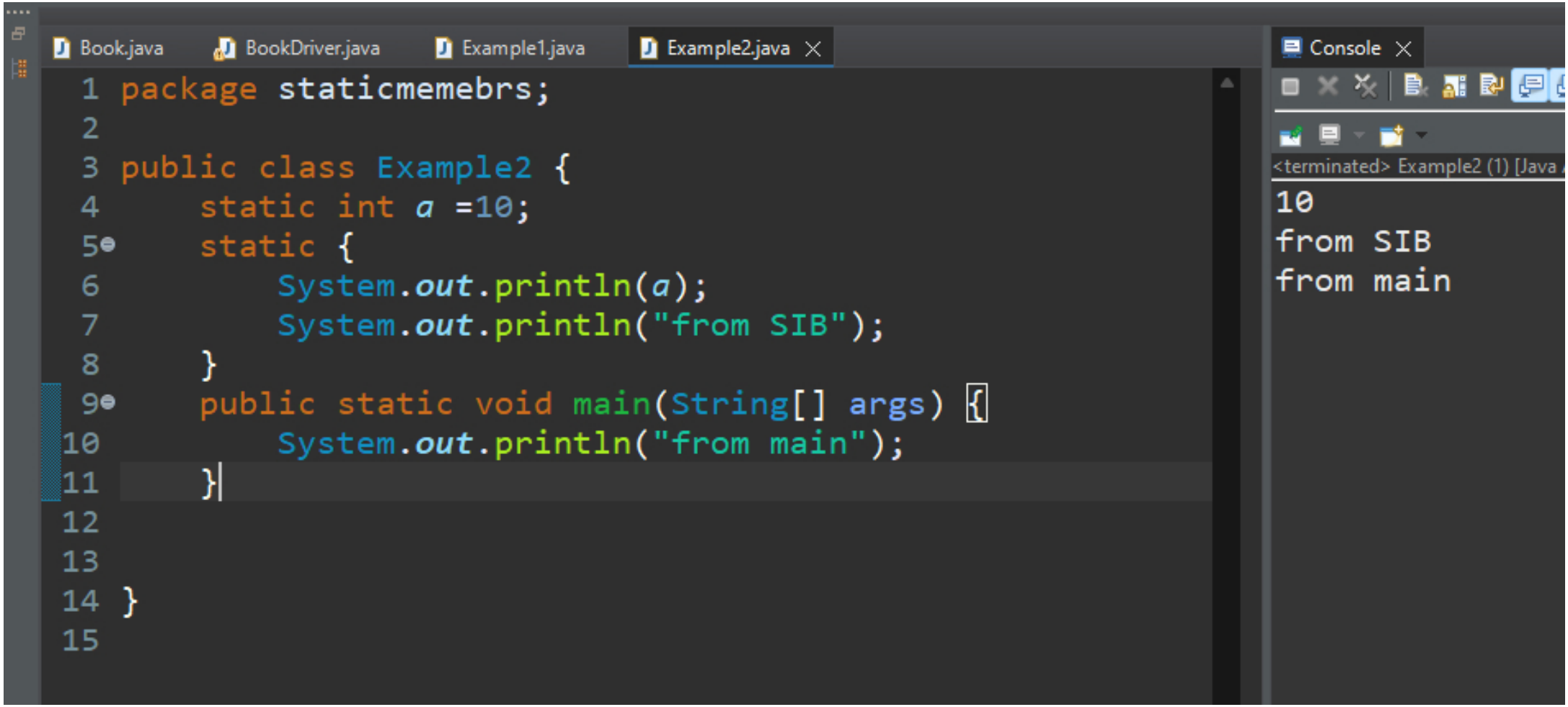
```
class Browser
{
static void search(String url)
{
----
----
}
static
{
search("google.com);
}
}
```

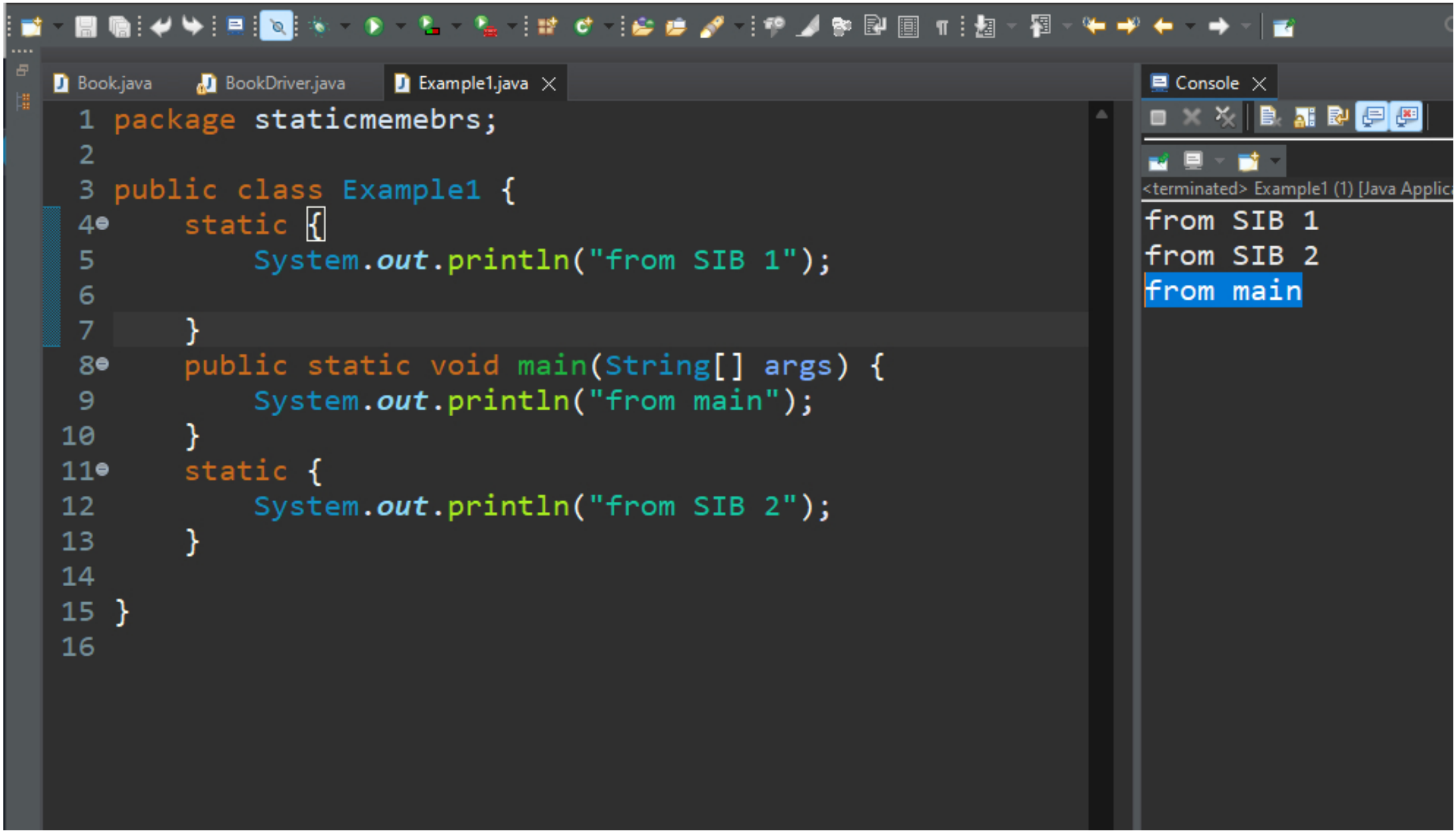
```
class BrowserDriver
{
static void search(String url)
{
Sopln(url + " is opened");
}
public static void main(String[] args)
{
Sopln("Browser is Launched");
search("instagram.com");
}
static {
search("www.google.com");
}
}
```

```
www.ggogle.com is
 opened
Browser is Launched
instagram.com is
opened
```

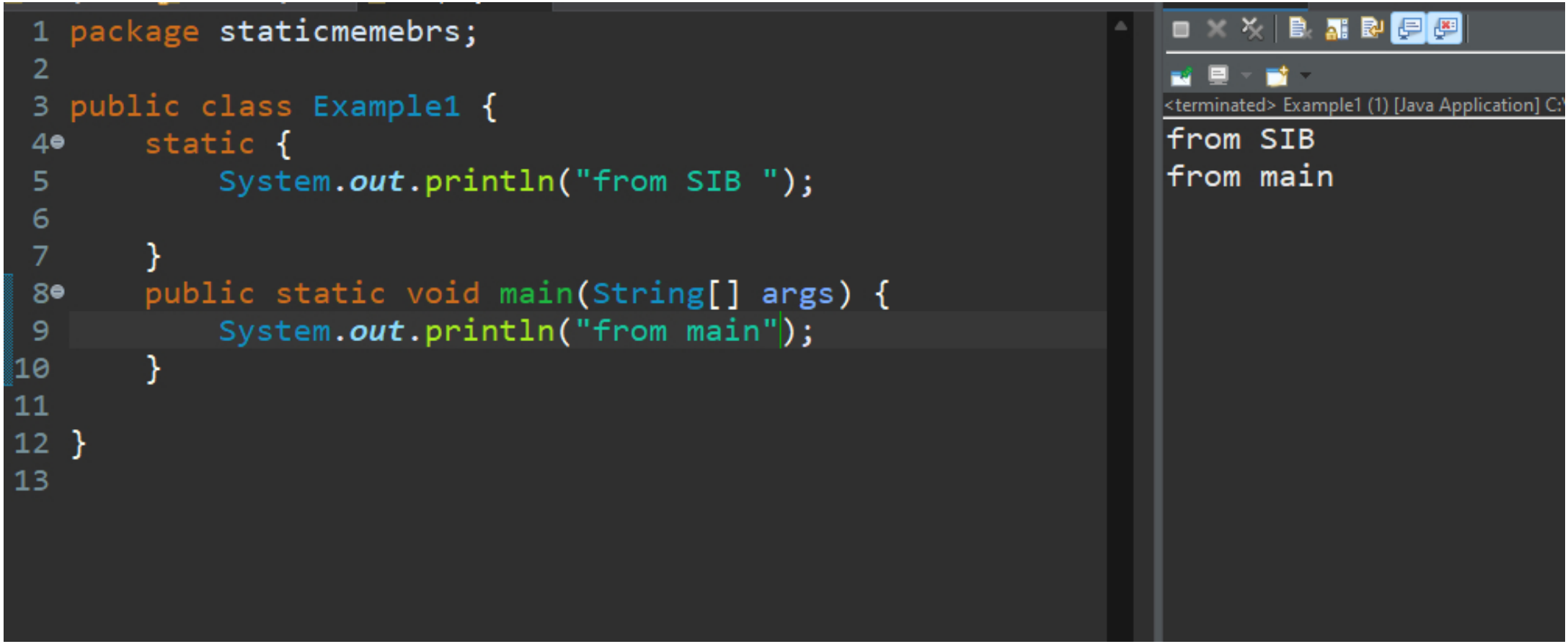

Book.java
BookDriver.java
Example1.java
Example2.java
BrowserDriver.java
package staticmemebrs;
public class BrowserDriver {
static void search(String url)
{
System.out.println(url + " is opened");
}
public static void main(String[] args)
{
System.out.println("Browser is Launched");
search("instagram.com");
}
static {
search("www.google.com");
}
}
Console
<terminated> BrowserDriver [Java Application] C:\Progra
www.google.com is opene
Browser is Launched
instagram.com is opened


eclipse-workspace - E142/src/staticmemebrs/Example2.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Book.java
BookDriver.java
Example1.java
Example2.java
1 package staticmemebrs;
2
3 public class Example2 {
4 static int a=test();
5 public static int test() {
6 System.out.println("from test");
7 return 10;
8 }
9 public static void main(String[] args) {
10 System.out.println("from main");
11 System.out.println(a);
12 }
13
14 }
15
Console
<terminated> Example2 (1) [Java Applicatio
from test
from main
10

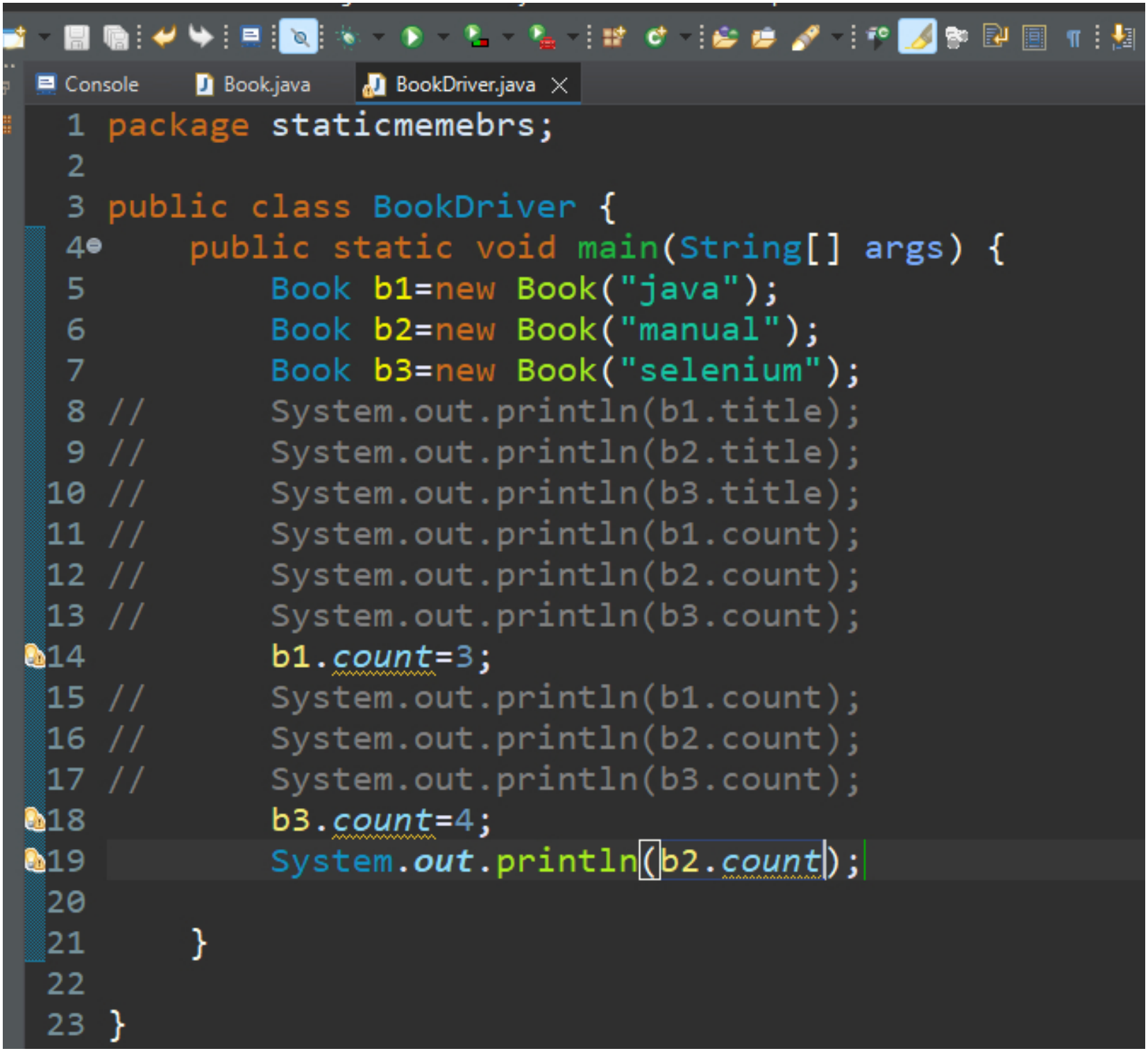

Book.java
BookDriver.java
Example1.java
Example2.java
1 package staticmemebrs;
2
3 public class Example2 {
4 static int a =10;
5 static {
6 System.out.println(a);
7 System.out.println("from SIB");
8 }
9 public static void main(String[] args) {
10 System.out.println("from main");
11 }
12
13
14 }
15
Console
<terminated> Example2 (1) [Java
10
from SIB
from main




```java
package staticmemebrs;

public class Example1 {
    static {
        System.out.println("from SIB 1");

    }
    public static void main(String[] args) {
        System.out.println("from main");
    }
    static {
        System.out.println("from SIB 2");
    }

}

```

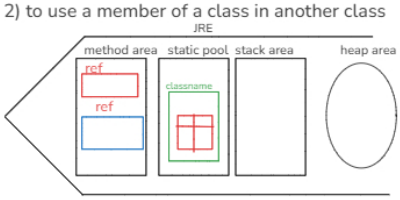

1 package staticmemebrs;
2
3 public class Example1 {
4 static {
5 System.out.println("from SIB ");
6
7 }
8 public static void main(String[] args) {
9 System.out.println("from main");
10 }
11
12 }
13
<terminated> Example1 (1) [Java Application] C:
from SIB
from main

```java
package staticmemebrs;


public class Book {
	String title;
	Book(String title){
		this.title=title;
	}
	static int count;


}

```


Console
Book.java
BookDriver.java
1 package staticmemebrs;
2
3 public class BookDriver {
4 public static void main(String[] args) {
5 Book b1=new Book("java");
6 Book b2=new Book("manual");
7 Book b3=new Book("selenium");
8 // System.out.println(b1.title);
9 // System.out.println(b2.title);
10 // System.out.println(b3.title);
11 // System.out.println(b1.count);
12 // System.out.println(b2.count);
13 // System.out.println(b3.count);
14 b1.count=3;
15 // System.out.println(b1.count);
16 // System.out.println(b2.count);
17 // System.out.println(b3.count);
18 b3.count=4;
19 System.out.println(b2.count);
20
21 }
22
23 }

Loading Process of a Class :

-->class is loaded from hardisk to JRE
for 2 purposes
1)to execute a class
2) to use a member of a class in another class



Step 1 : Whenever we load a class into JRE , inside class static area one block will be created for the class and the name of the class is given to that block
-->whatever is present inside that block can be accessed with the help of classname

Step 2 : All the members of the class i.e., static variables, static methods,static blocks are loaded one by one into JRE in top to bottom order

static methods :
the entire block will be loaded into the method area or code segment area and reference will be generated and this reference is copied into block which is present in static pool area

static variables :
memory will be allocated inside the same block where class block is created and it is assigned with default values

static initalizer :

all of them one by one will be copied into the method area some reference will be generated
-->for static block as a programer we dont give any name, JVM will give a default name and the generated reference will be copied into the class block which is created in the class static area

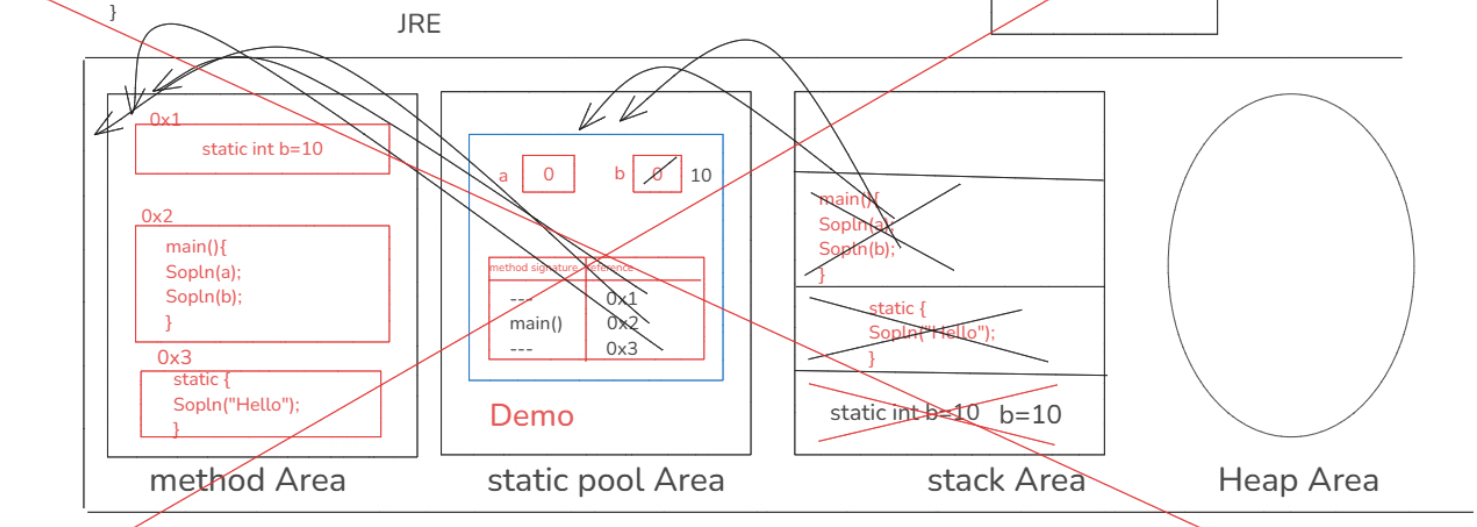
Step 3: the static initalizers will get executed
-->if it is not present then the 3rd step will not happen

```
Ex : class Demo{
       static int a;
         static int b=10;
       public static void main(String[] args)
       {
        Sopln(a);
        Sopln(b);
     }
      static{
        Sopln("Hello");
      }
}
```

static variable
static initializer

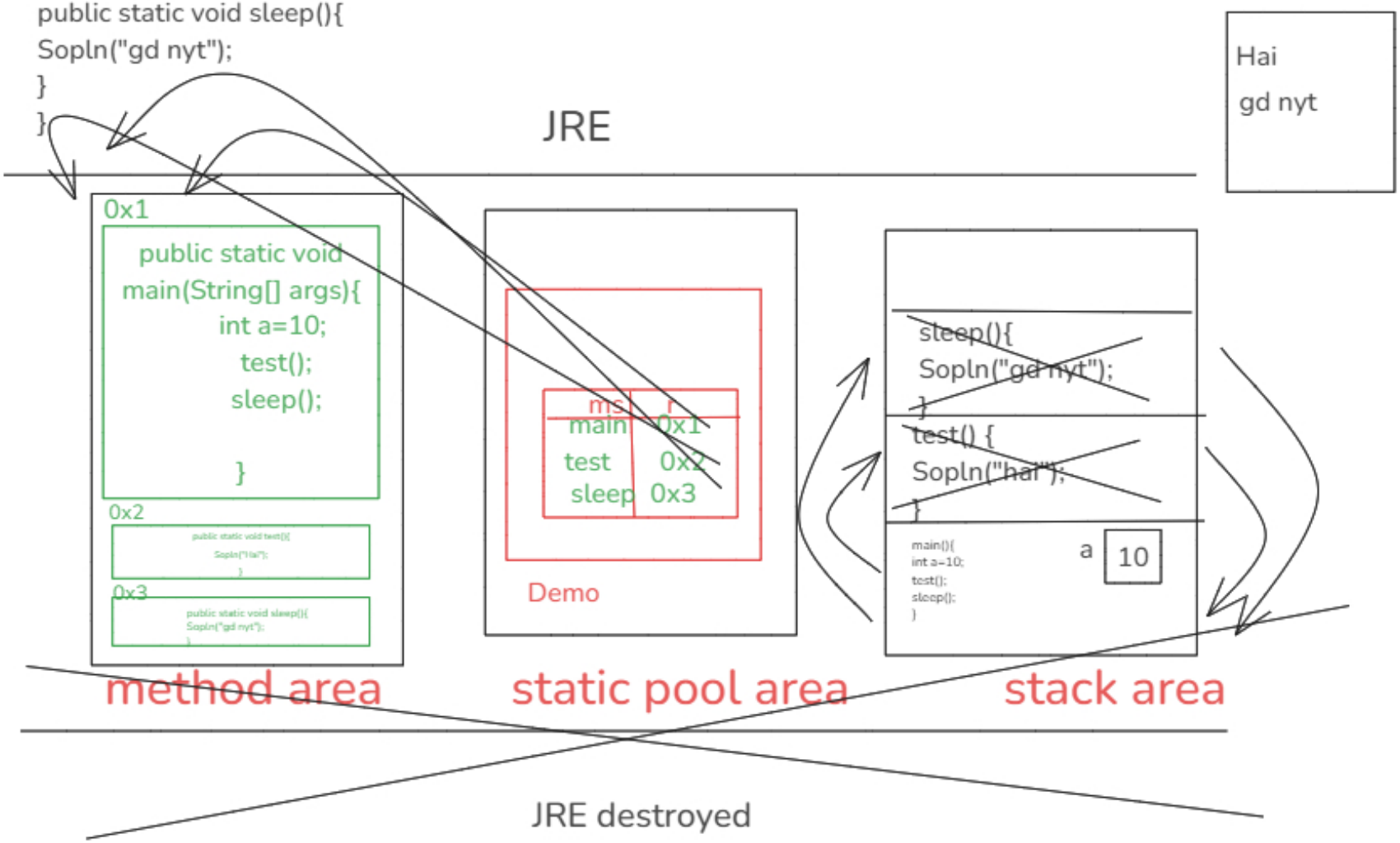
Hello
0
10

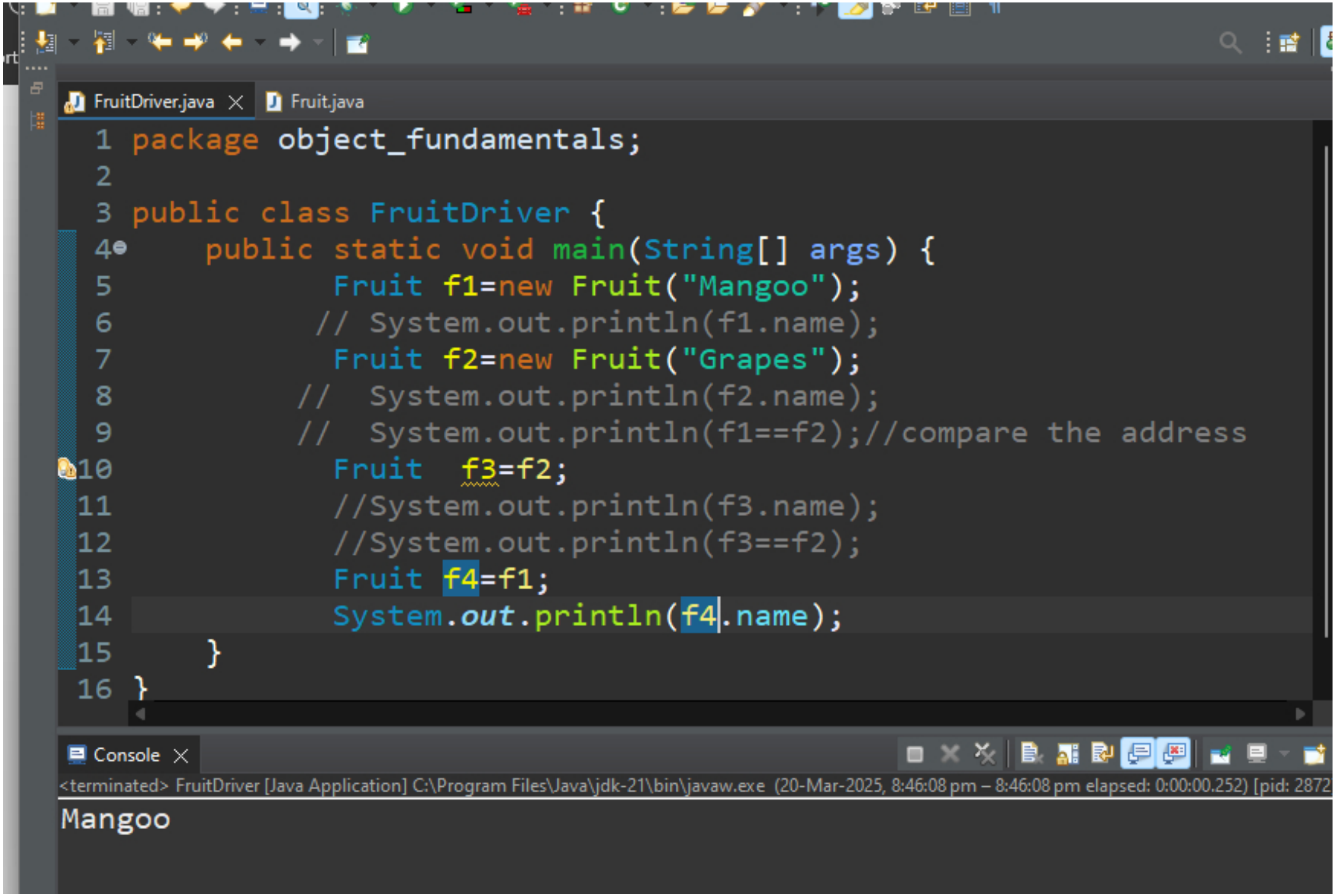
Hello
0
10



-->after complete execution of java application , the entire JRE gets destroyed

```
Ex : class Demo{
        public static void main(String[] args){
         int a=10;
          test();
          sleep();

}
public static void test(){
Sopln("Hai");
}
public static void sleep(){
Sopln("gd nyt");
}
}
```




FruitDriver.java
Fruit.java
1 package object_fundamentals;
2
3 public class FruitDriver {
4 public static void main(String[] args) {
5 Fruit f1=new Fruit("Mangoo");
6 // System.out.println(f1.name);
7 Fruit f2=new Fruit("Grapes");
8 // System.out.println(f2.name);
9 // System.out.println(f1==f2);//compare the address
10 Fruit f3=f2;
11 //System.out.println(f3.name);
12 //System.out.println(f3==f2);
13 Fruit f4=f1;
14 System.out.println(f4.name);
15 }
16 }
Console
<terminated> FruitDriver [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (20-Mar-2025, 8:46:08 pm – 8:46:08 pm elapsed: 0:00:00.252) [pid: 2872
Mangoo

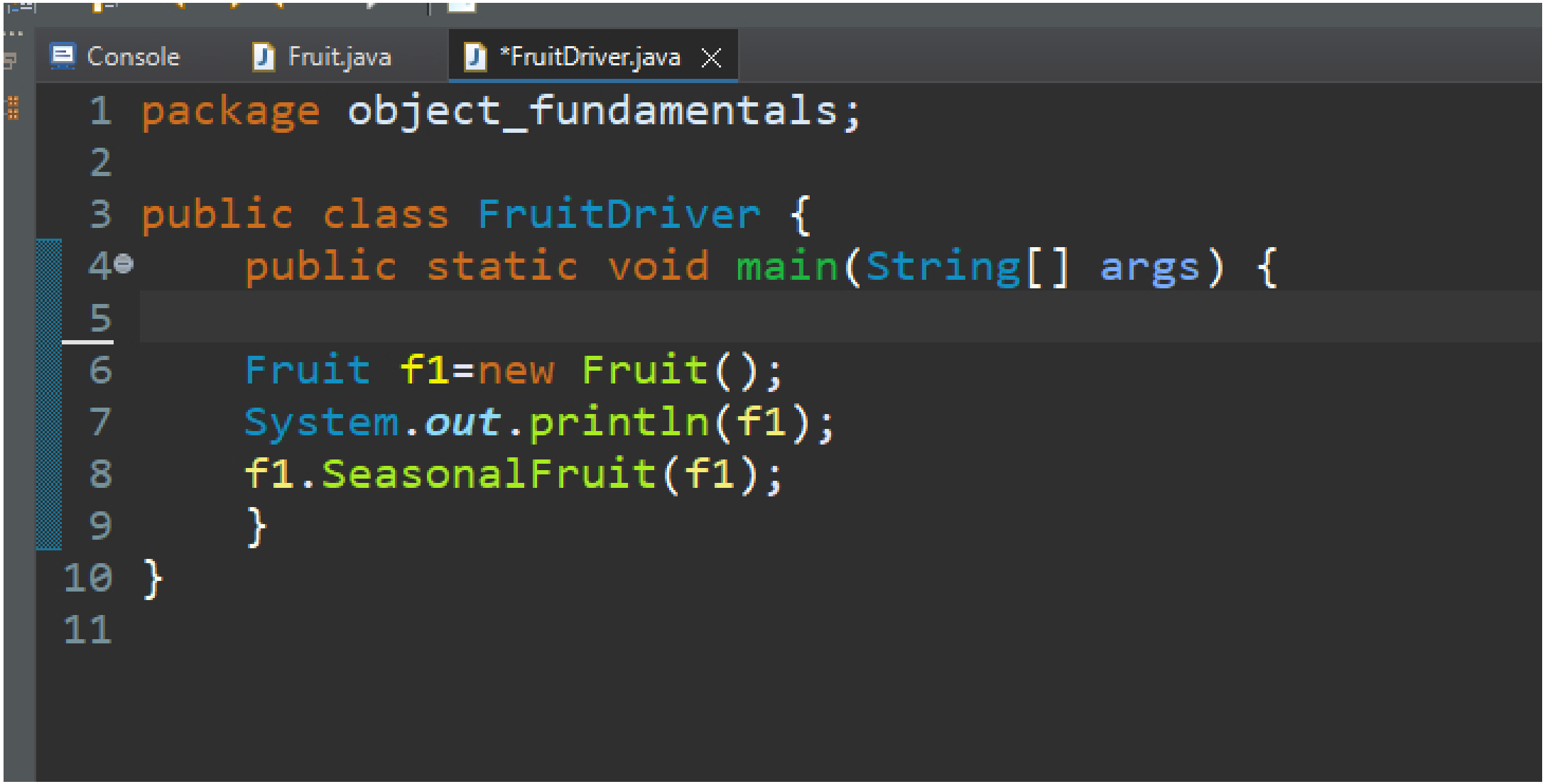
Console

<terminated> FruitDriver [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (20-Mar-2025, 8:38:56 pm – 8:38:56 pm elapsed: 0:00

```
object_fundamentals.Fruit@372f7a8d
object_fundamentals.Fruit@372f7a8d
```

Fruit.java | FruitDriver.java

```
1 package object_fundamentals;
2
3 public class Fruit {
4       public void SeasonalFruit(Fruit f) {
5           System.out.println(f);
6       }
7 }
8
```

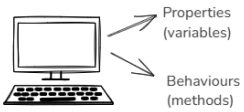

Console
Fruit.java
*FruitDriver.java
1 package object_fundamentals;
2
3 public class FruitDriver {
4 public static void main(String[] args) {
5
6 Fruit f1=new Fruit();
7 System.out.println(f1);
8 f1.SeasonalFruit(f1);
9 }
10 }
11

There are 4 Object Oriented Principles :

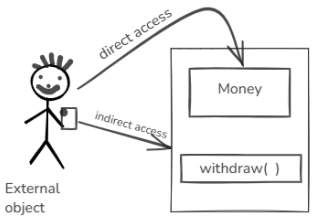
1)Encapsulation
2)Inheritance
3)Polymorphism
4)Abstraction

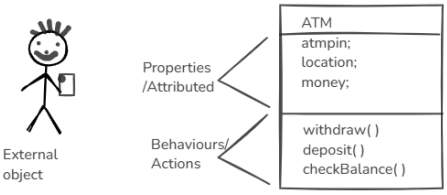
ENCAPSULATION

-->The Process of Wrapping The Attributes(Properties) and Actions(Behaviours) of an Object together is known as Encapsulation



Ex : Teller machine



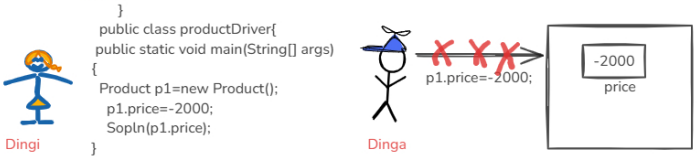


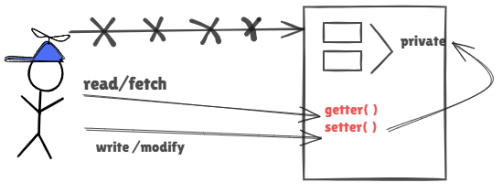
DataHiding : The process of restricting the direct access to the Properties/Attributes of the object and providing controlled and restricted access to the properties of an object is Known as DataHiding

--> secure the data
-->we need to validate the data before storing it

--> to secure the data we need to use a keyword called private .
-->private is a Access Modifier
-->it will restrict the external access

```
Ex : class Product
     {
private double price ;
     }
 public class productDriver{
 public static void main(String[] args)
{
  Product p1=new Product();
   p1.price=-2000;
   Sopln(p1.price);
}
}
```





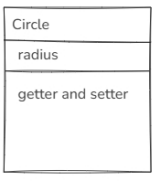
getter( ) - it is used to fetch private data

setter( ) - it is used to modify the private data

```
Ex : class Product
     {
      private double price;
      //setter method
       void setPrice(double price){
      if(price>0){
       this.price=price;
    }
      //getter method
       double getPrice(){
        return this.price;
        }
  }
```

```
class ProductDriver
{
public static void main(String[] args)
{
product p1=new Product();
//store -2000 in price
  p1.setPrice(-2000);
  Sopln(p1.getPrice());

  }
  }
```

Task :



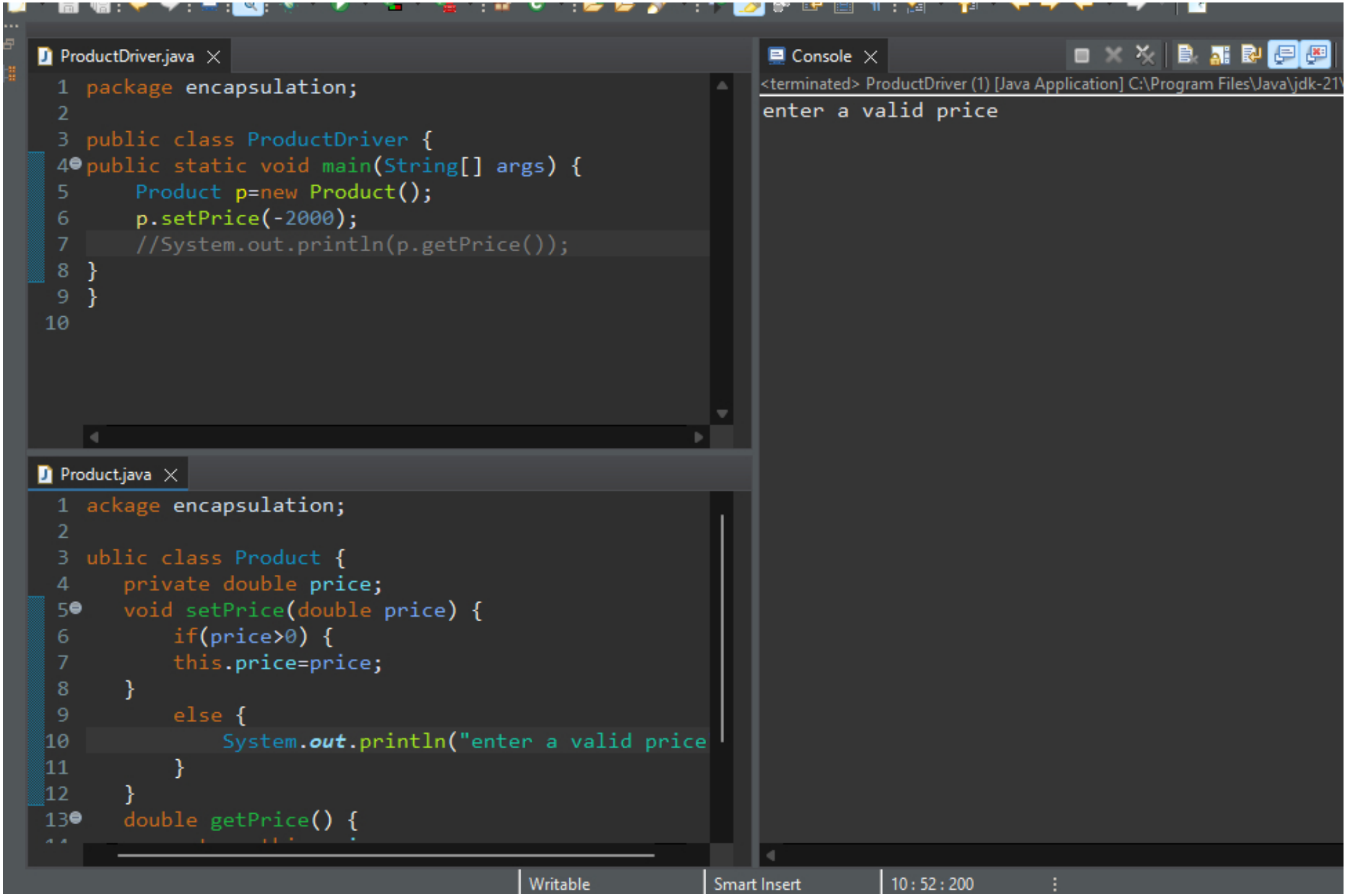
Driver
3 circle objects
pass the radius
print the radius

Console | *Student.java

```java
package encapsulation;

public class Student {
    private String name;
    private long phno;
    //getter for name
    public String getName() {
        return name;
    }
    //setter name variable
    public void setName(String name) {
        this.name = name;
    }
    //getter for phno
    public long getPhno() {
        return phno;
    }
    //setter for phno
    public void setPhno(long phno) {
        if(phno>0) {
        this.phno = phno;
    }
        else {
            System.out.println("enter a valid phon
        }
    }


}
```

*StudentDriver.java

```java
package encapsulation;

public class StudentDriver {
    public static void main(String[] args) {
        Student s=new Student();
        s.setName("Harika");
        System.out.println(s.getName());
        s.setPhno(98765432111);
        System.out.println(s.getPhno());
    }

}
```


ProductDriver.java
1 package encapsulation;
2
3 public class ProductDriver {
4 public static void main(String[] args) {
5 Product p=new Product();
6 p.setPrice(-2000);
7 //System.out.println(p.getPrice());
8 }
9 }
10
Product.java
1 ackage encapsulation;
2
3 ublic class Product {
4 private double price;
5 void setPrice(double price) {
6 if(price>0) {
7 this.price=price;
8 }
9 else {
10 System.out.println("enter a valid price
11 }
12 }
13 double getPrice() {
Console
<terminated> ProductDriver (1) [Java Application] C:\Program Files\Java\jdk-21
enter a valid price
Writable
Smart Insert
10 : 52 : 200

ProductDriver.java

```java
package encapsulation;

public class ProductDriver {
public static void main(String[] args) {
    Product p=new Product();
    p.setPrice(2000);
    System.out.println(p.getPrice());
}
}

```

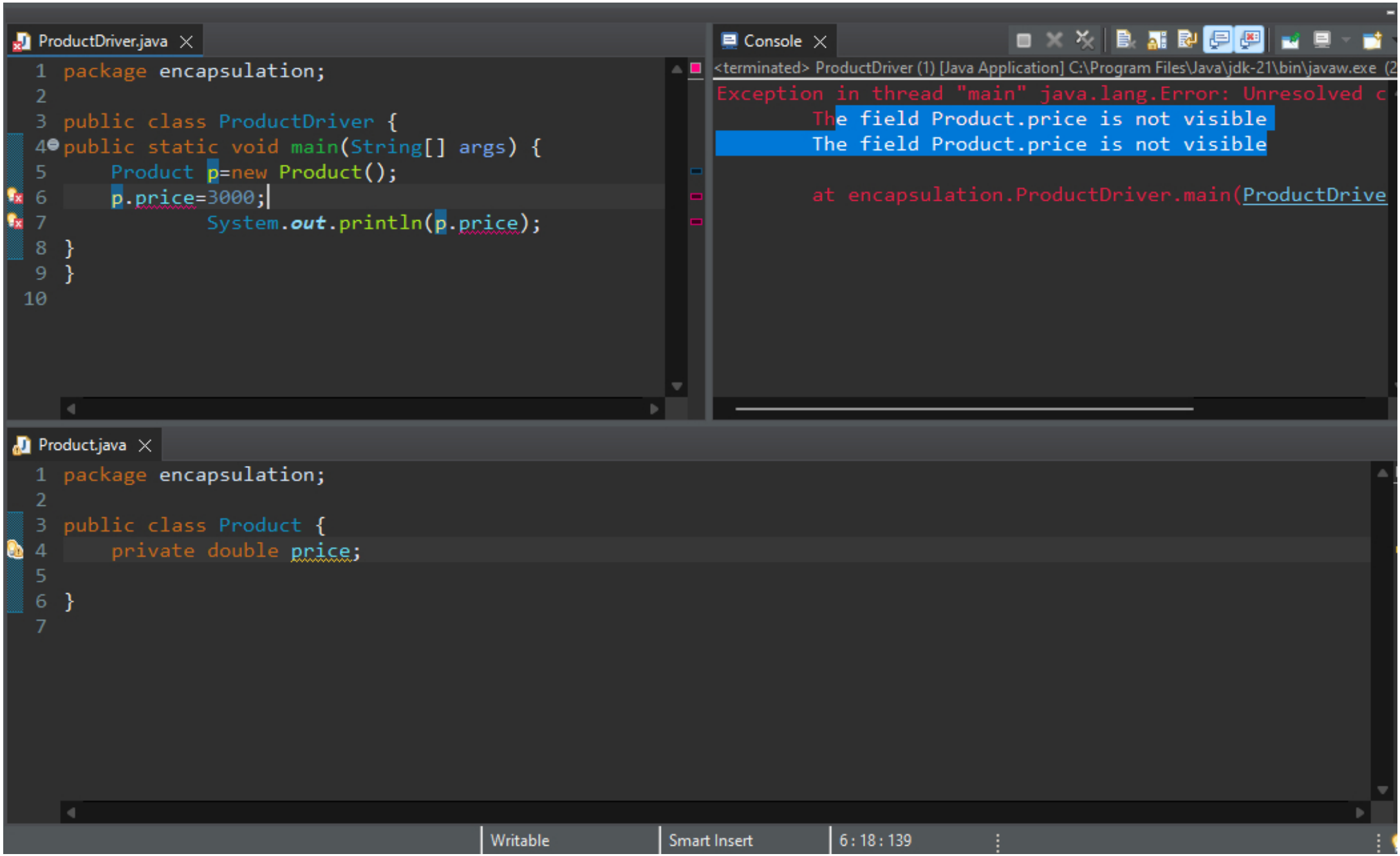
Console

<terminated> ProductDriver (1) [Java Application] C:\Program Fi

```
2000.0
```

Product.java

```java
package encapsulation;

public class Product {
    private double price;
    void setPrice(double price) {
        this.price=price;
    }
    double getPrice() {
        return this.price;
    }

}
```


ProductDriver.java
1 package encapsulation;
2
3 public class ProductDriver {
4 public static void main(String[] args) {
5 Product p=new Product();
6 p.price=3000;
7 System.out.println(p.price);
8 }
9 }
10
Console
<terminated> ProductDriver (1) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (2
Exception in thread "main" java.lang.Error: Unresolved c
The field Product.price is not visible
The field Product.price is not visible
at encapsulation.ProductDriver.main(ProductDrive
Product.java
1 package encapsulation;
2
3 public class Product {
4 private double price;
5
6 }
7
Writable
Smart Insert
6 : 18 : 139

CircleDriver.java | Circle.java | CirclesDriver.java | Circles.java

```java
package encapsulation;

public class Circle {
    private double radius;
    //setter
    public void setRadius(double radius) {
        this.radius=radius;
    }
    //getter
    public double getRadius()
    {
        return radius;
    }
    //constructor(no-arg)
    Circle(){

    }
    //constructor(parameterized)
    Circle(double radius){
        this.radius=radius;
    }
}

```

CircleDriver.java | Circle.java | CirclesDriver.java | Circles.java

```java
package encapsulation;

public class CircleDriver {
    public static void main(String[] args) {
        Circle c1=new Circle(5);
        System.out.println(c1.getRadius());
        Circle c2=new Circle();
        c2.setRadius(5);
        System.out.println(c2.getRadius());


    }

}

```

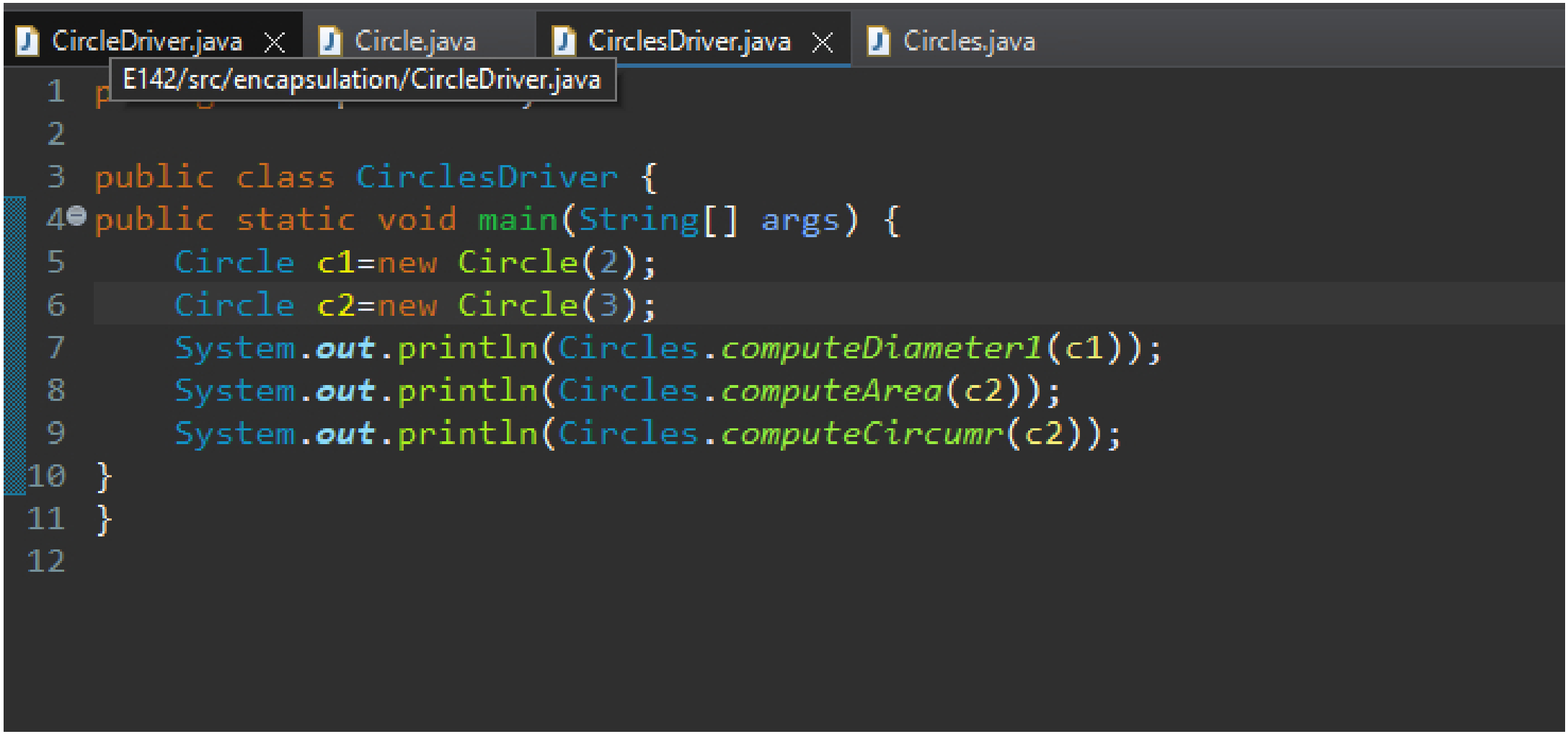
CircleDriver.java | Circle.java | CirclesDriver.java | Circles.java

```java
package encapsulation;

public class Circles {
    //static methods to perform task
    //1.to calculate Diameter
    public static double computeDiameter1(Circle c) {
        return 2*c.getRadius();

    }

    //2.to calculate Area
     public static double computeArea(Circle c ) {
        return (22.0/7.0)*c.getRadius()*c.getRadius();
    }

     //3.to calculate Circumference
     public static double  computeCircumr(Circle c) {
         return 2*(22.0/7.0)*c.getRadius();


    }

}

```

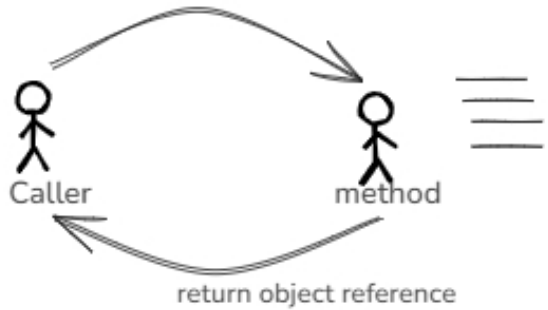
Console

terminated> Test (2) [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (24-Mar-2025, 8:48:25 pm – 8:48:26 pm elaps

.0


CircleDriver.java
Circle.java
CirclesDriver.java
Circles.java
E142/src/encapsulation/CircleDriver.java
1
2
3 public class CirclesDriver {
4 public static void main(String[] args) {
5 Circle c1=new Circle(2);
6 Circle c2=new Circle(3);
7 System.out.println(Circles.computeDiameter1(c1));
8 System.out.println(Circles.computeArea(c2));
9 System.out.println(Circles.computeCircumr(c2));
10 }
11 }
12

Method returning Object reference :



Syntax :

[modifiers] returntype name(Formal args..)

{

non-primitive type

return obj_ref;

}

Ex :

```
class Box{
      double length;
       Box(double length){
       this.length=length;
       }
public static Box createBox(double length){
   return new Box(length);
}
```

factory method

```
class BoxDriver{
public static void main(string[] args){
//Box b1=new Box(5);
//Box b2=new Box(3);
  Box b1=Box.createBox(4);
   Sopln(b1.length);
   Sopln(Box.createBox(3).length);
}
}
```

instead of this i want to design a method which creates box objects

return reference.length

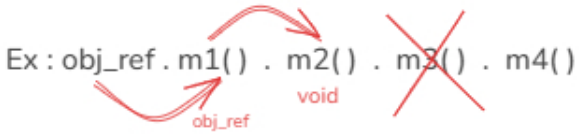
-->factory method produces the object and gives you the reference of the object.

-->return type should be non-primitive type

Method Chaining :

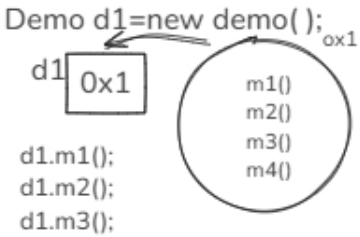
The process of calling / invoking multiple methods one after another in a single statement with the help of . operator is known as Method Chaining.

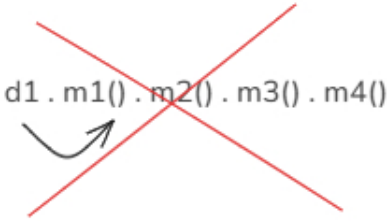
obj_ref.m1( ).m2( ).m3( ).m4( )



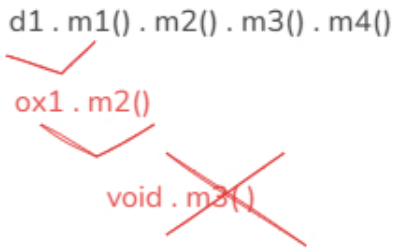
-->we cannot achieve method chaining if method return type is void.
-->we cannot achieve method chaining if method return type is primitive.
-->we can achieve this by making the return type as non-primitive / object reference

```
Ex  : class Demo{
        void m1( ){ }
        void m2( ){ }
        void m3( ){ }
        void m4( ) { }
      }
```

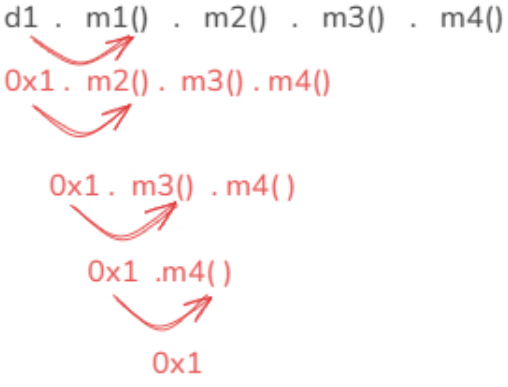




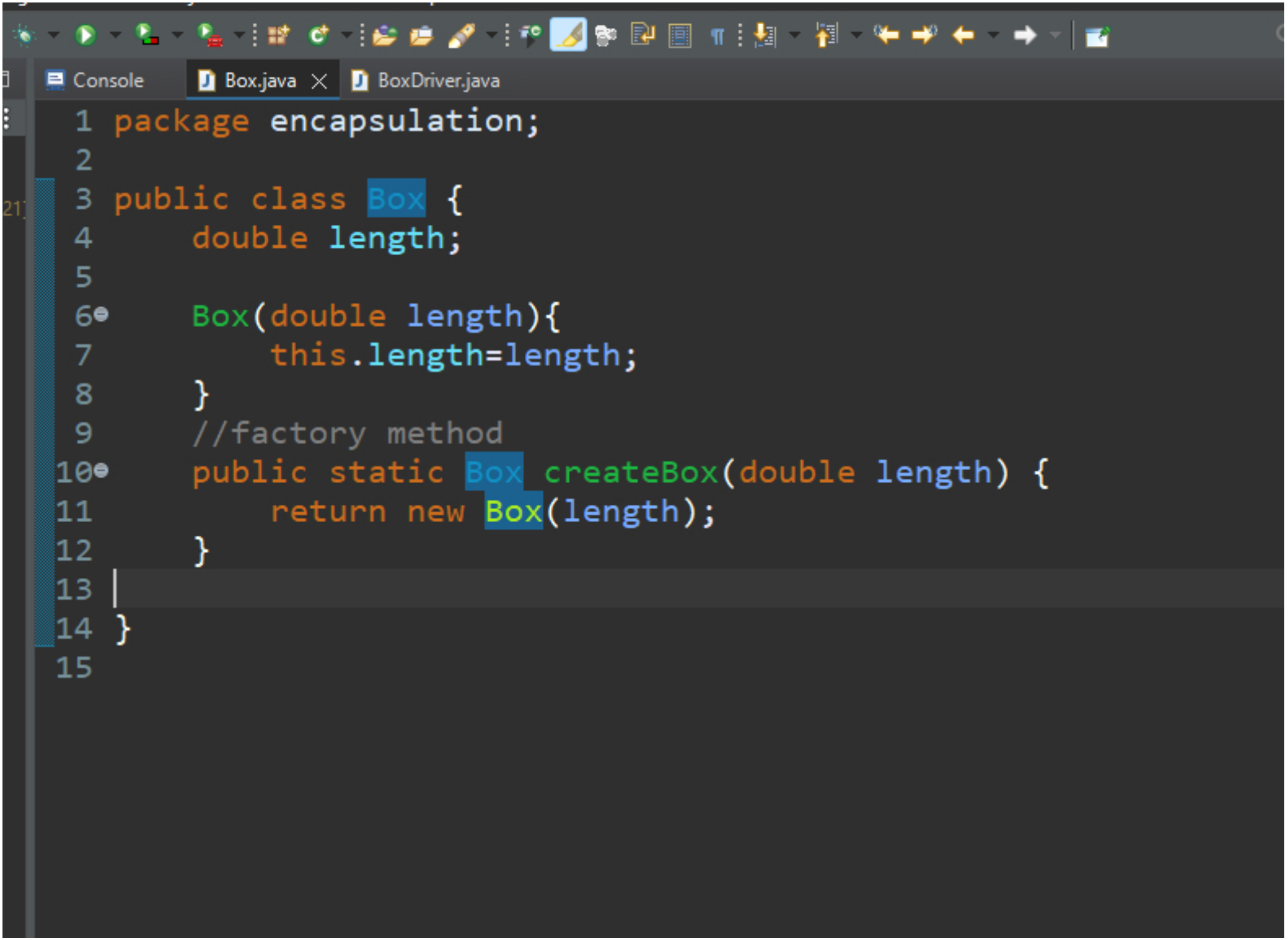

```
Ex :  class Demo{
        Demo m1( ){
         return this;
       }
       void m2(){}
       void m3(){}
       void m4(){}
       }
```

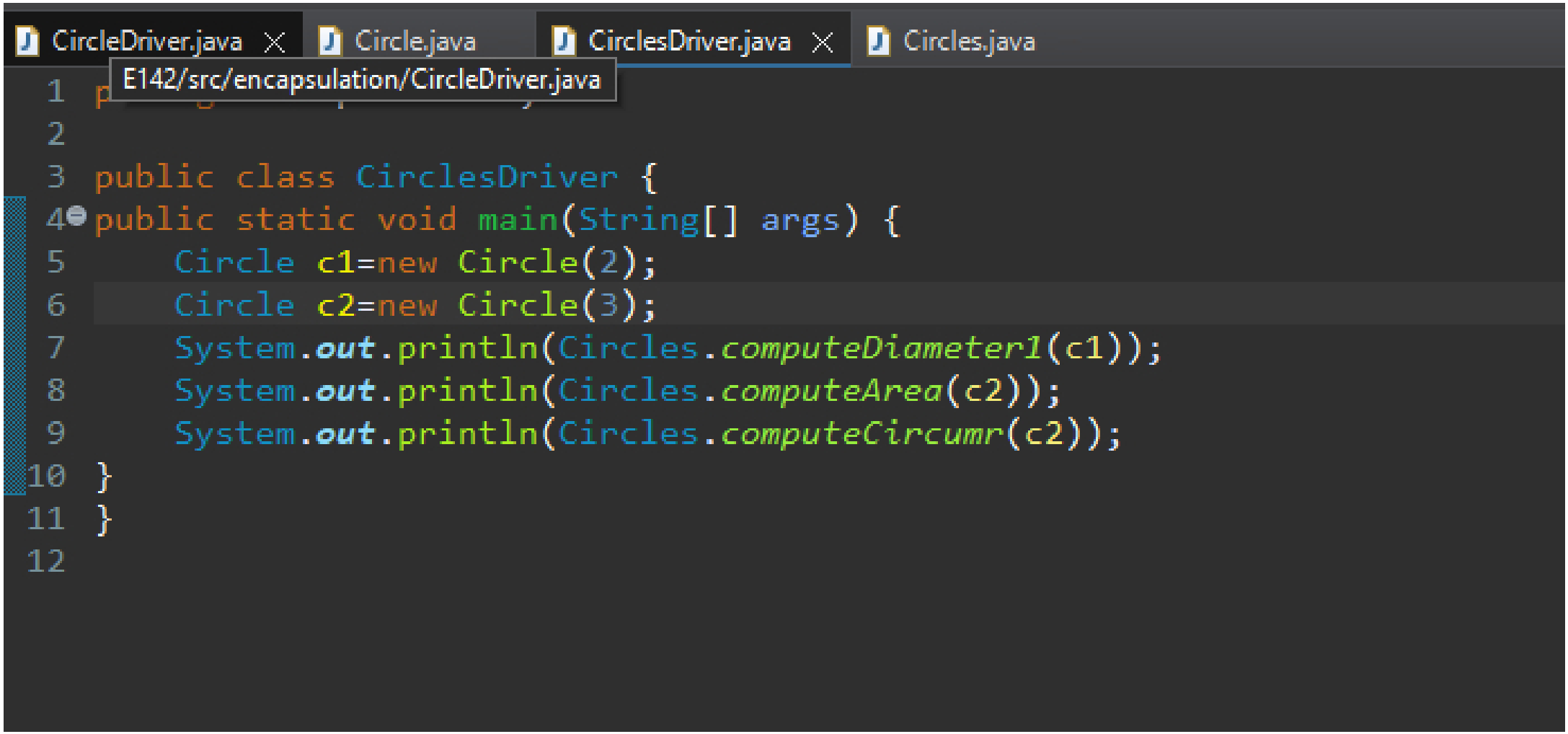



```
Ex :  class Demo{
        Demo m1(){
        return this;
        }
        Demo m2(){
        return this;
        }
        Demo m3(){
        return this;
        }
       Demo m4(){
        return this;
       }
  }
```

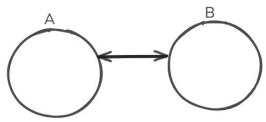


```java
package encapsulation;

public class BoxDriver {
public static void main(String[] args) {
	Box b1=Box.createBox(3);
	System.out.println(b1.length);
	System.out.println(Box.createBox(5).length);
}
}

```


Console
Box.java
BoxDriver.java
1 package encapsulation;
2
3 public class Box {
4 double length;
5
6 Box(double length){
7 this.length=length;
8 }
9 //factory method
10 public static Box createBox(double length) {
11 return new Box(length);
12 }
13
14 }
15


CircleDriver.java
Circle.java
CirclesDriver.java
Circles.java
E142/src/encapsulation/CircleDriver.java
1
2
3 public class CirclesDriver {
4 public static void main(String[] args) {
5 Circle c1=new Circle(2);
6 Circle c2=new Circle(3);
7 System.out.println(Circles.computeDiameter1(c1));
8 System.out.println(Circles.computeArea(c2));
9 System.out.println(Circles.computeCircumr(c2));
10 }
11 }
12

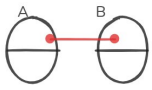
Relationship :



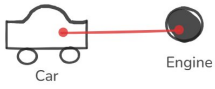
-->one object is having connection with another object
is known as Relationship
-->it is classified into 2 types
1)Has-A Relationship
2)Is-A Relationship

1)Has-A Relationship :

An Object being dependent on another Object is called as Has-A Relationship.

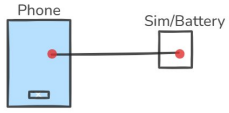


A is completely dependend on B Object

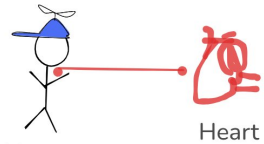


Car object is dependend on engine object

Car has-a Engine



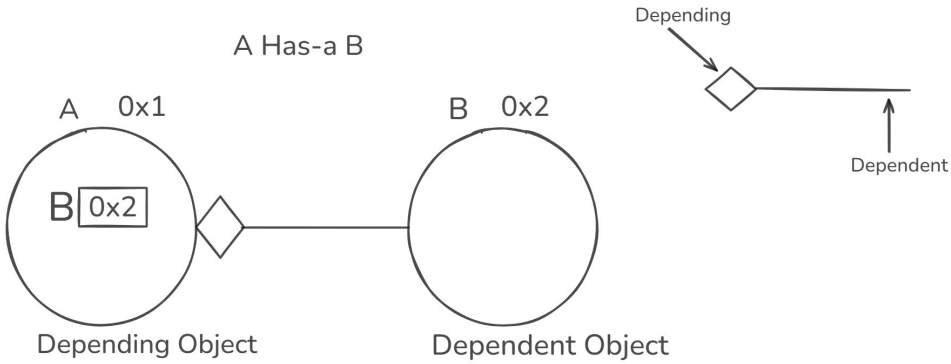
Phone has-a Battery



Human has-a Heart

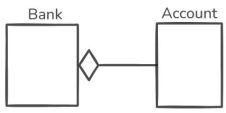
Examples :

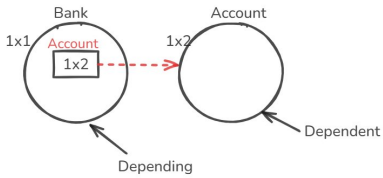
Employee has-a job
Gun has-a Bullet
Bank has-a Account .....etc



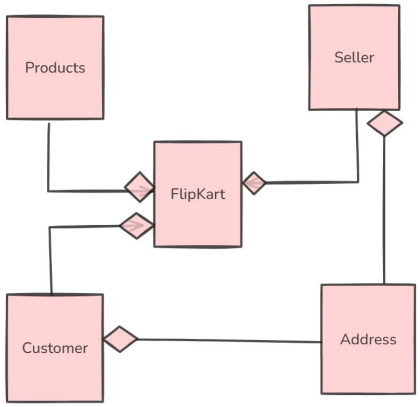
Examples :

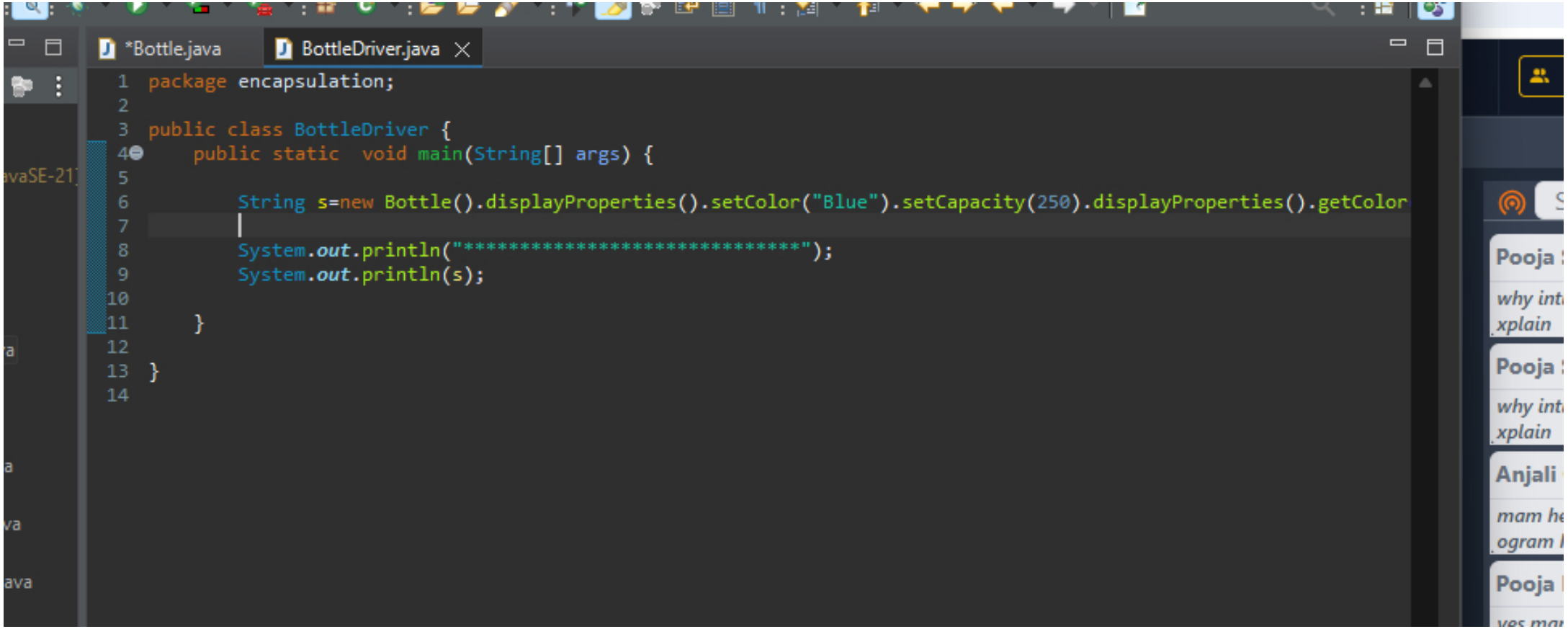
Bank and Account

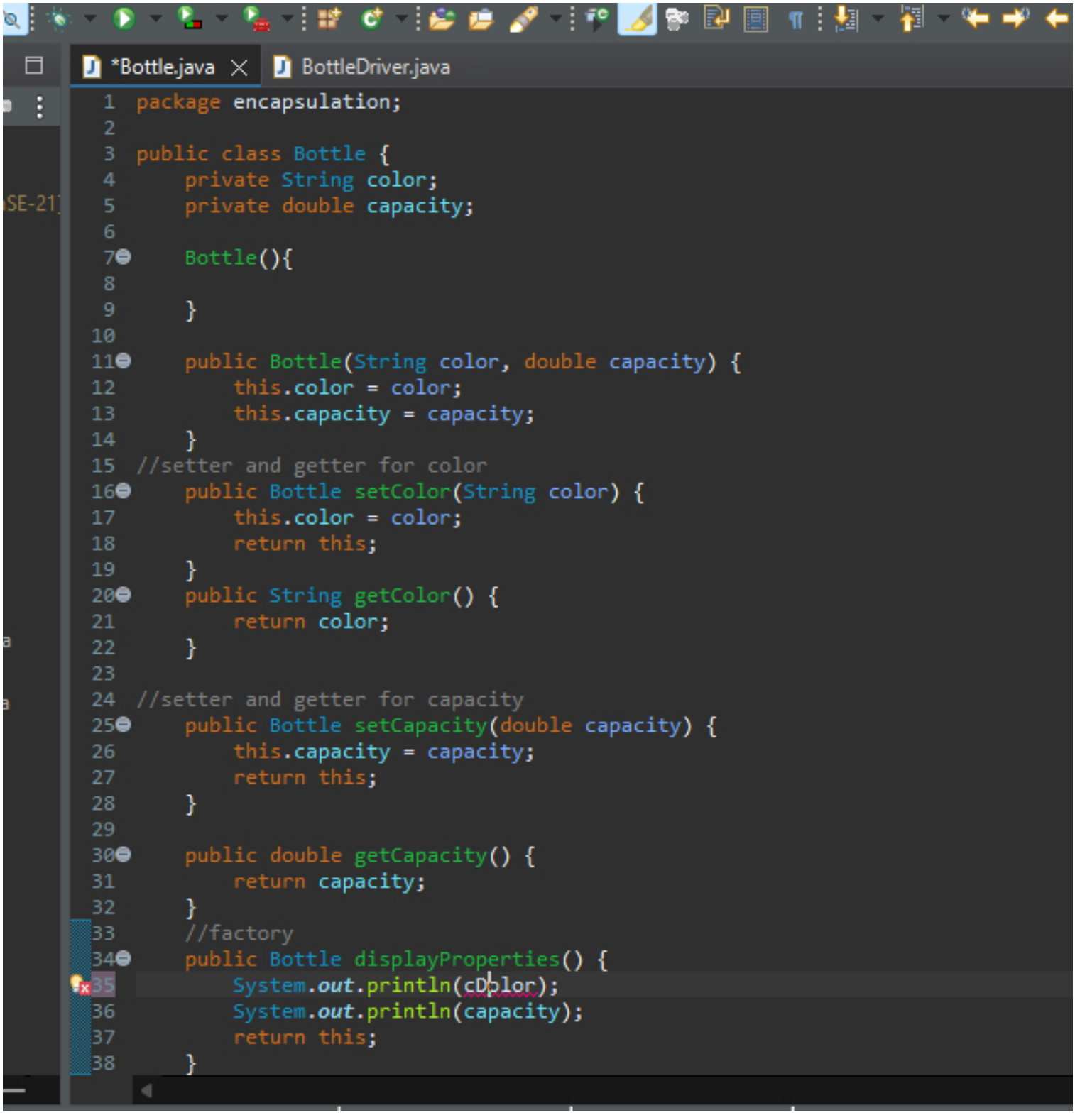




***

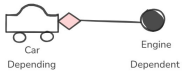


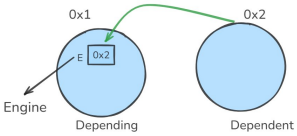

*Bottle.java
BottleDriver.java
1 package encapsulation;
2
3 public class BottleDriver {
4 public static void main(String[] args) {
5
6 String s=new Bottle().displayProperties().setColor("Blue").setCapacity(250).displayProperties().getColor
7
8 System.out.println("*******************************");
9 System.out.println(s);
10
11 }
12
13 }
14
Pooja
why int
xplain
Pooja
why int
xplain
Anjali
mam he
ogram
Pooja


*Bottle.java
BottleDriver.java
1 package encapsulation;
2
3 public class Bottle {
4 private String color;
5 private double capacity;
6
7 Bottle(){
8
9 }
10
11 public Bottle(String color, double capacity) {
12 this.color = color;
13 this.capacity = capacity;
14 }
15 //setter and getter for color
16 public Bottle setColor(String color) {
17 this.color = color;
18 return this;
19 }
20 public String getColor() {
21 return color;
22 }
23
24 //setter and getter for capacity
25 public Bottle setCapacity(double capacity) {
26 this.capacity = capacity;
27 return this;
28 }
29
30 public double getCapacity() {
31 return capacity;
32 }
33 //factory
34 public Bottle displayProperties() {
35 System.out.println(cDblor);
36 System.out.println(capacity);
37 return this;
38 }

To Implement Has-A Relationship :

Car and Engine
Car Has-A Engine





S1 : Create a Reference variable of Dependent Object inside the Depending Object

Ex : Create variable of Engine type inside Car Object

S2 : Create an Instance(Object) of Dependent Object and copy the reference to the Depending Object

```
* class Car {
        Engine e;
  }
```

S 1.1 : Design the Bluprint for the Dependent Object

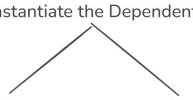
```
class Engine {

}
```

S 1.2 : Create a non-static Variable in Depending Class of Dependent Type

```
Class Car{
      Engine e ;
}
```

Design Technique To Create Instance of Dependent Object

S 2 : To Instantiate the Dependent Object



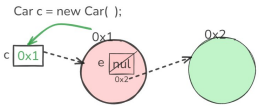
type 1	type 2
Early Instantiation	Lazy Instantiation
-->we achieve by Initializer	we achieve by Helper method
-->The Dependent object is Created Immediately after the Depending Object	-->The Dependent object is created Later , it is not created when Depending object is Created

Type 1 :
Early Instantiation :

```
class Engine {

}
```

```
class Car{
    Engine e =new Engine( );
}
```

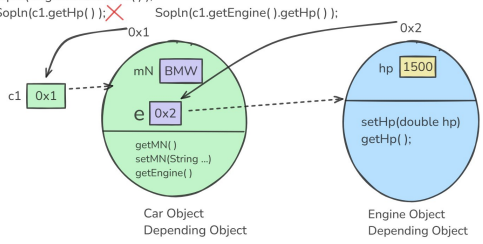
non-static Initializer




```
Ex :  class Engine {
        private double hp;
      void setHp(double hp){
      this.hp=hp;
      }
       double getHp( ){
       return hp;
       }
       Engine( ){
      }
      Engine(double hp){
         setHp(hp);
      }
```

```
class Car {
private String modelName;
void setModelName(String modelName){
this.modelName=modelName;
}
String getModelName( ){
return modelName;
}
private Engine e = new Engine(1500);

  //getter
     Engine getEngine( ){
       return e;
  }
  Car( ){
  }
  Car(String ModelName){
  setModelName(modelName);
  }
  }
```

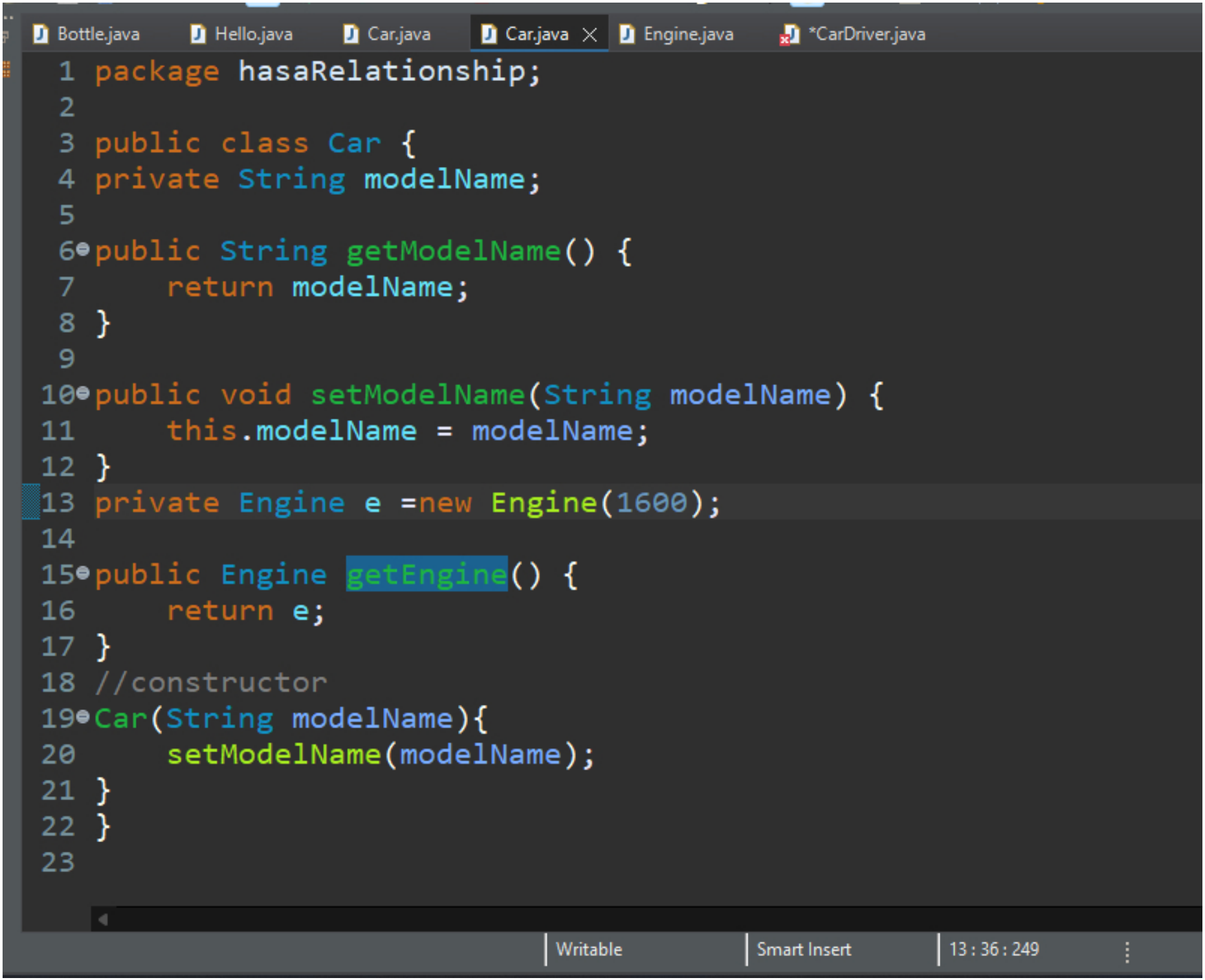
```
class CarDriver{
    public static void main(String[] args){
    //create an instance of car
      Car c1=new Car("BMW");
     //print the car model
      Sopln(c1.getModelname( ) );
       Sopln(c1.getHp( ) );
       Sopln(c1.getEngine( ).getHp( ) );
}
}
```

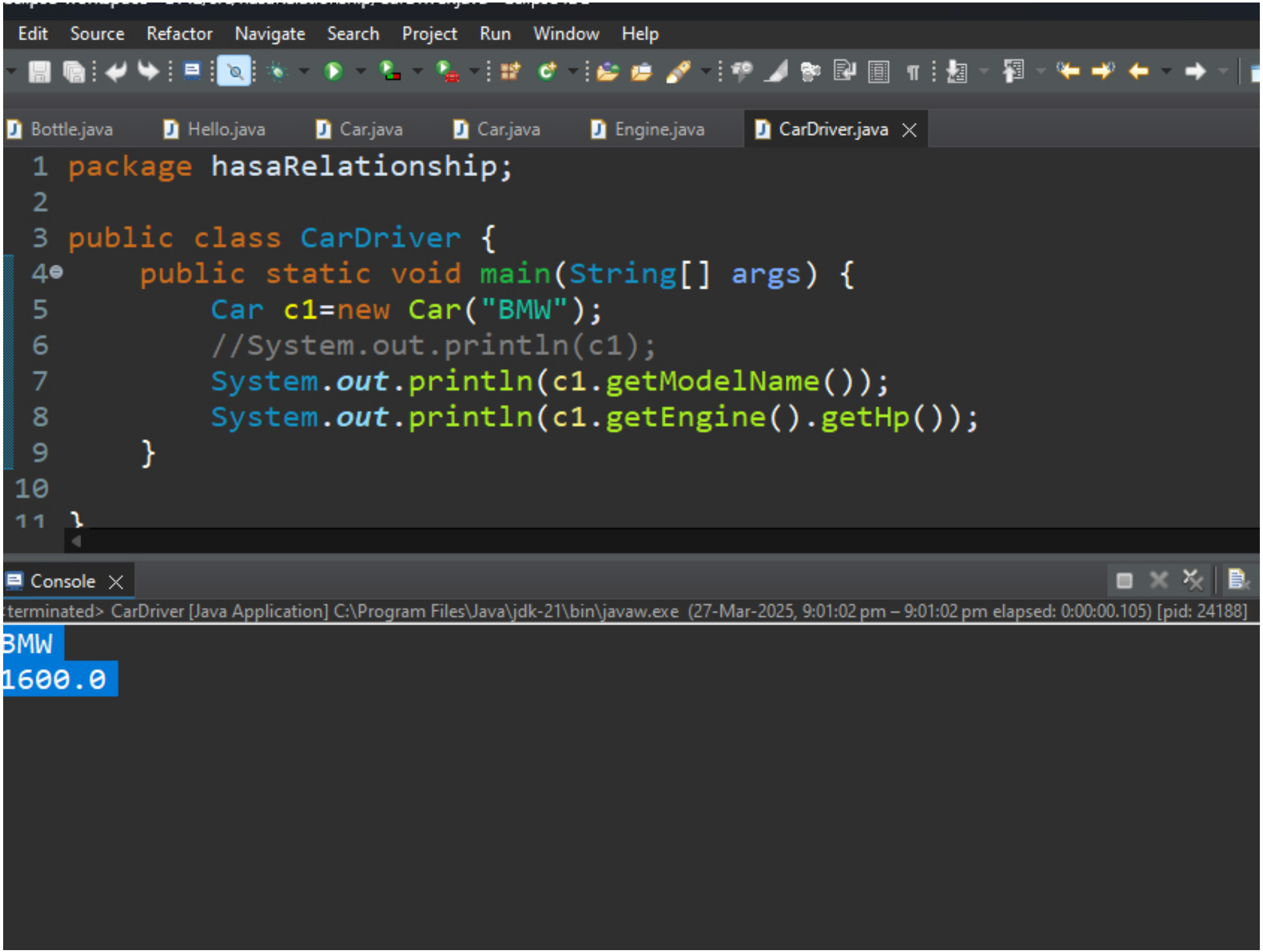


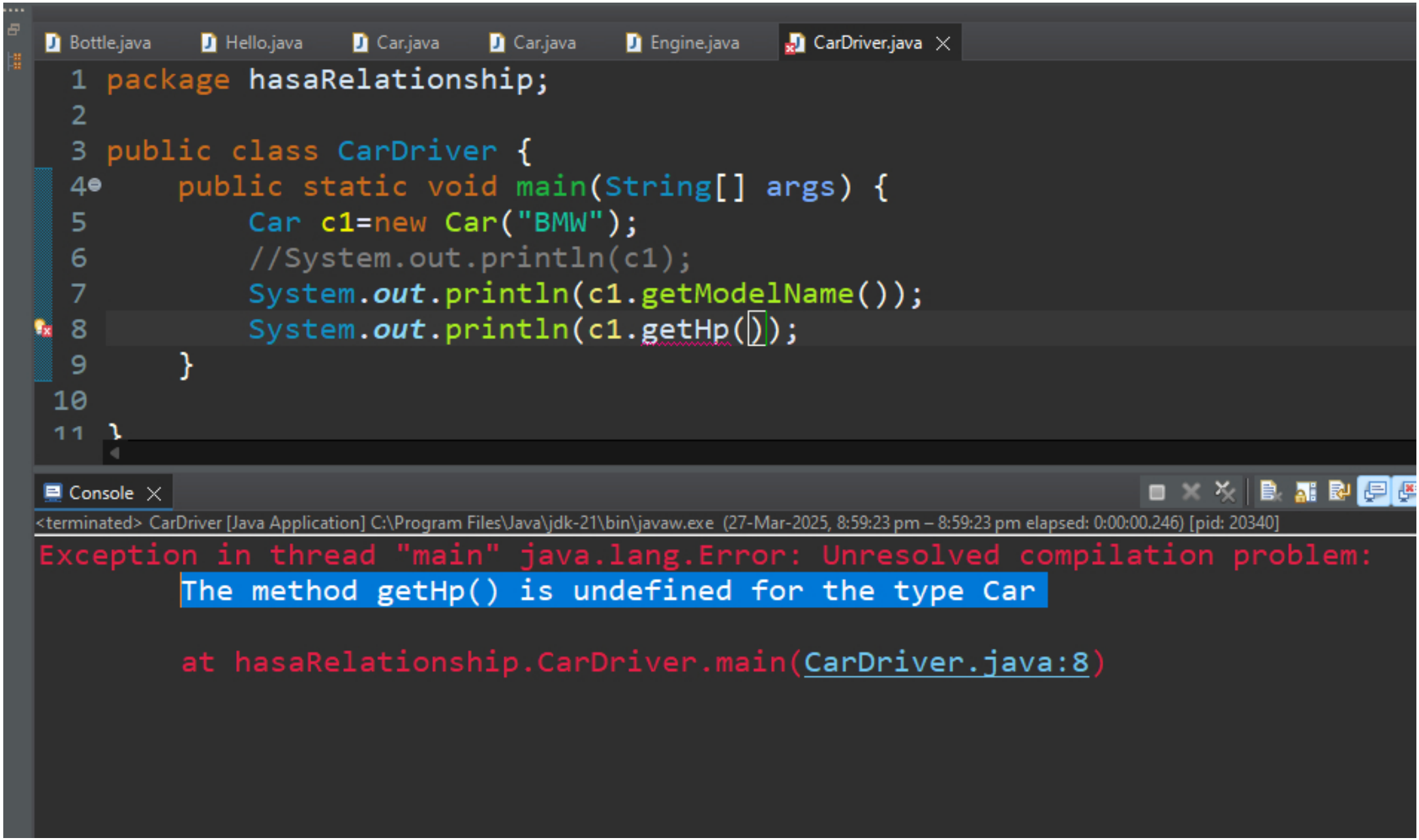
Bottle.java | Hello.java | Car.java | Car.java | Engine.java | *CarDriver.java

```java
package hasaRelationship;

public class Engine {
private double hp;

public double getHp() {
    return hp;
}

public void setHp(double hp) {
    this.hp = hp;
}

Engine(){

}
 Engine(double hp) {
    setHp(hp);
}

}

```

Writable | Smart Insert | 18 : 15

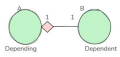

Bottle.java
Hello.java
Car.java
Car.java
Engine.java
*CarDriver.java
1 package hasaRelationship;
2
3 public class Car {
4 private String modelName;
5
6 public String getModelName() {
7 return modelName;
8 }
9
10 public void setModelName(String modelName) {
11 this.modelName = modelName;
12 }
13 private Engine e =new Engine(1600);
14
15 public Engine getEngine() {
16 return e;
17 }
18 //constructor
19 Car(String modelName){
20 setModelName(modelName);
21 }
22 }
23
Writable
Smart Insert
13 : 36 : 249


Edit Source Refactor Navigate Search Project Run Window Help
Bottle.java Hello.java Car.java Car.java Engine.java CarDriver.java
1 package hasaRelationship;
2
3 public class CarDriver {
4 public static void main(String[] args) {
5 Car c1=new Car("BMW");
6 //System.out.println(c1);
7 System.out.println(c1.getModelName());
8 System.out.println(c1.getEngine().getHp());
9 }
10
11 }
Console
terminated> CarDriver [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (27-Mar-2025, 9:01:02 pm – 9:01:02 pm elapsed: 0:00:00.105) [pid: 24188]
BMW
1600.0

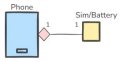

Bottle.java
Hello.java
Car.java
Car.java
Engine.java
CarDriver.java
1 package hasaRelationship;
2
3 public class CarDriver {
4 public static void main(String[] args) {
5 Car c1=new Car("BMW");
6 //System.out.println(c1);
7 System.out.println(c1.getModelName());
8 System.out.println(c1.getHp());
9 }
10
11 }
Console
<terminated> CarDriver [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (27-Mar-2025, 8:59:23 pm – 8:59:23 pm elapsed: 0:00:00.246) [pid: 20340]
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
The method getHp() is undefined for the type Car
at hasaRelationship.CarDriver.main(CarDriver.java:8)

Multiplicty :

1) one-to-one Relationship :



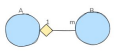
Ex :



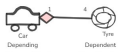

```
class Battery
{

}
```

```
class Phone
{
Battery b;
}
```

2)one-to-many Relationship :

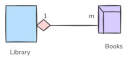


Ex :

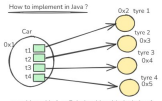


One car object becomes a Container for 4 tyre objects

Ex :

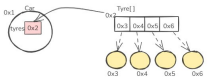


How to implement in Java ?

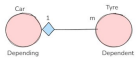


to achieve this 1-m Relationship with the help of Arrays (or) Collection Framework.

we should not create 4 variables , we can create an Array



How to design a class to achieve one-to-many Relationship



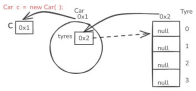
S1 : Design Dependent class

```
class Tyre {

}
```

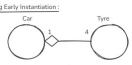
S 1.2 : Design Depending class

```
class Car{
Tyre[ ] tyres = new Tyre[ 4 ];

}
```

Array Object is used to store the Address of tyres



Using Early Instantiation :




```
class Tyre {
  String brand ;
Tyre(String Brand){
this.brand=brand;
}
}
```

```
class Car {
Tyre[] tyres={new Tyre("MRF"),
              new Tyre("Apollo"),
              new Tyre("Birla"),
              new Tyre("Ceat")};
}
```

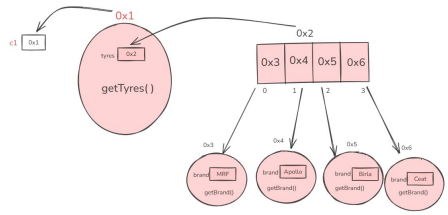
```
class CarDriver{
p s v m (S[] args){
 Car c=new Car( );
 //display all the brands
}
}
```

```
Ex:   Dependent Class

public class Tyre {
    private String brand;

    String getBrand() {
        return brand;
    }

    void setBrand(String brand) {
        this.brand = brand;
    }

    Tyre(String brand){
        setBrand(brand);
    }

}
```

```
Depending Class

public class Carr {
private Tyre[] tyres= {new Tyre("MRF"),new Tyre("Apollo"),
                       new Tyre("Birla"),new Tyre("Ceat")};

Syntax method
    Tyre[]  getTyres(){
        return tyres;
    }
}
```

Driver Class

```
public class Driver {
 public static void main(String[] args) {
Carr c1=new Carr();
 System.out.println(c1);
  System.out.println(c1.getTyres().length);
//print brand of all tyres
 for(int i=0;i<c1.getTyres().length;i++) {
System.out.println(c1.getTyres()[i].getBrand());
 }
 }
 }
```



Task :

Depending Class

College
- name setters and getters constructor Student Array getter for Student Array

Dependent Class

Student
- name -id setters and getters constructor

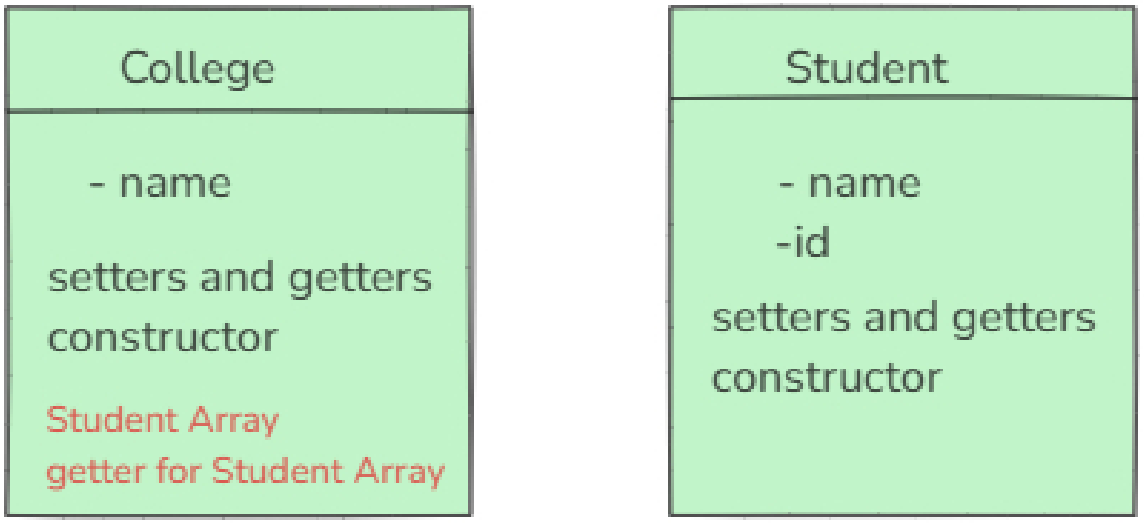
3 students

Driver Class College Object
print the student names print the student id's

Task :

Depending Class

Dependent Class



3 students

Driver Class
College Object

print the student names
print the student id's