This documentation is for reference purpose only and is for those who have attended the classroom sessions at Thinking Machines.

- During your classroom session appropriate theory needs to be written against each example.
- You are required to bring this book daily for your classroom sessions.
- Some examples won't compile. They have been written to explain some rules.
- If you try to understand the examples without attending theory sessions then may god help you.

S.No.	Topic	Page
1	C++ - Two techniques of creating objects	
2	Encapsulation	
3	Polymorphism	
4	C++ Inheritance - Visibility mode : public	
5	C++ Inheritance - Visibility mode : private	
6	Inheritance	
7	C++ Multiple Inheritance and problems associated with it	
8	C++ Virtual Inheritance	
9	Static method	
10	Static property	
11	C++ - Garbage value	
12	Rules of assignment in context to a local variable	
13	Local variable as final	
14	Method overriding	
15	Final method	
16	Final class	
17	Rules of assignment in context to a Object variable	
18	C++ - Assigning value to class member	
19	System.out.println	
20	Constructors	
21	super keyword to invoke base class constructor	
22	super keyword to access base class member	
23	initializer block	
24	static initializer block	
25	Object variable as final	
26	Class variable as final	
27	Java is a strongly typed language	
28	Base class reference variable can store reference of a derived class object	
29	C++ - Virtual function	
30	Virtual Polymorphism	
31	abstract method and abstract class	
32	Promotion	

S.No.	Topic	Page
33	Donkey – Monkey Wala Example	
34	Object is the super most class in java	
35	C++ - Unsafe Code	
36	Array is treated as an object	
37	Resizing array	
38	Array of reference variables	
39	String	
40	println with Object as parameter and toString method	
41	Deep comparison	
42	Lexical comparison	
43	Exception handling	
44	try block with multiple catch blocks	
45	nested try catch	
46	catch with super class reference variable	
47	Creating and throwing custom exceptions	
48	Caught – Uncaught Exceptions and throws keyword	
49	try – catch - finally	
50	Parsing String to int	
51	Integer.parseInt	
52	Command line arguments	
53	Introduction to factory class	
54		
55		
56		
57		

System.out.Print(" ");
- System — Class
- out — Pointer / reference variable
- Print — function

Print – print करता line change नहीं करता

Println – line को भी change करता है

Q) Primitive Data type in Java [इसमें 8 type है]

A) long, int, short, byte, double, float, char and boolean

~~They are following eight (8) type~~

Kalu K; → class/interface/enum

int j;

double d;

K [ ] → variable
can store address of an obj. of type Kalu → pointer

d [ ] → variable
can store double type value

j [ ] → variable
can store int type value

अगर इन 8 type में से है तो variable अगर ना हो तो Pointer.

Java में हमेशा name less object होंगे

double cote में हमेशा string होते है "string"

number + number → Arithmetic operation perform होगा

"string" + "string" → concatenation होगा

अगर एक string है तो —
"string" + a
a + "string"
→ इसमें जो string नहीं है उसे string में convert करेगा फिर concatenation

"Hello" + 10 + 20

[Hello1020] Hello का zero है zero जोड़ेंगे

Entry Point Function
Example 1

```
class psp
{
public static void main(String gg[])
{
System.out.print("Hello");
System.out.println("God is");
System.out.println("great");
}
}
```

Hello
God is
great

Java PSP ↵
JVM
(Java virtual machi

(Two techniques of creating objects)
C++

```
#include<iostream.h>
class Bulb
{
private:
int w;
public:
void setWattage(int e)
{
w=e;
}
int getWattage()
{
return w;
}
};
void main()
{
Bulb b;
b.setWattage(60);
cout<<b.getWattage()<<endl;
Bulb *p;
p=new Bulb;
p->setWattage(60);
cout<<p->getWattage()<<endl;
}
```

Encapsulation
Example 2

```
class Bulb
{
private int w;
public void setWattage(int e)
{
if(e>=0 && e<=240)
{
w=e;
}
else
{
```

difference d/w — Java, .net

Compiler | Interpreter

Compiler

① Compiler पूरे code को read करके एक file देता है

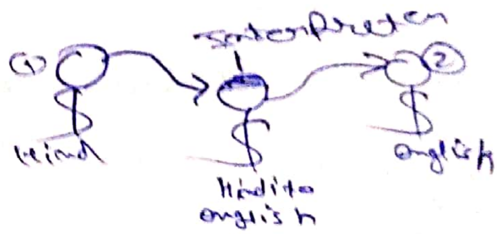
② C/C++ are Based on Compiler

Interpreter

① Interpreter code को पढ़कर (read) करके file अपने पास ही रखता है और execute कर देता है

② HTML, python, Node.js are Based on Interpreter

③



Interpreter contains Just in time compiler (JIT) — कुछ time के लिए compile होता

difference d/w ⇒

Top Down approach | Bottom up approach

Top Down approach

पहले Application बनाते है और इसके लिए module बनाना Top down approach कहते है

पहले हम पहले पूरा बनाकर फिर module बनाना Bottom up है Top Down नहीं

Bottom up approach

पहले application के module बनाना और module को मिलाकर application बनाना Bottom up approach कहते है।

difference procedural (programming) | Object oriented

Procedural

① पहले function class के बाहर लिखते है तो procedural programming

② C is based on procedural

③ C++ are based on both

Object oriented

① पहले class के अन्दर function लिखा तो object oriented programming

② Java, .net is based on ob. oriented

```
w=0;
}
}
public int getWattage()
{
return w;
}
}
class eg2psp
{
public static void main(String gg[])
{
Bulb b;
b=new Bulb();
b.setWattage(-30);
System.out.println("Wattage is "+b.getWattage());
Bulb k;
k=new Bulb();
k.setWattage(100);
System.out.println("Wattage is "+k.getWattage());
Bulb r;
r=null;
System.out.println("Ujjain");
r.setWattage(40);
System.out.println("Indore");
}
}
```

Wattage is -30
Wattage is 100
Ujjain
Exception in thread "main"
java.lang.NullPointerException
at eg2psp.main(eg1.java:35)

Example 3
Polymorphism

```
class Calculator
{
public void add(int e,int f,int g)
{
System.out.println("Total is "+(e+f+g));
}
public void add(int e,int f)
{
System.out.println("Total is "+(e+f));
}
}
class eg3psp
{
public static void main(String gg[])
{
Calculator c=new Calculator();
c.add(10,40);
c.add(30,40,50);
}
}
```

Total is 50
Total is 120

more same signature different (parameter)

---

abc.c

```
#include<stdio.h>
int main()
{
printf("Cool");
return 0;
}
```

gcc abc.c -o pqr.exe
↓
produces
pqr.exe

Q.) उस abc.c में कितने function define करे है?
Ans) एक function (main)

Q.) pqr.exe में कितने function define हो होंगे?
Ans) दो function (main, printf), अगर printf में किसी function को call किया जाएगा उसे भी count करेंगे।
→ ये दो function C-के compiler ने लिखे है।
→ ये main function compiler ने abc.c से देख कर लिखा है!

Q.) printf function compiler ने कहा से देख कर लिखा है
A) Library file से देख कर लिखा है।

) process of picking up function & putting in executable file.
Ans) Compile time linking

Q.) अगर code compile होने के बाद हमने header file, library file और compiler Delete कर दिया तो यह code execute होगा?
Ans) हाँ होगा! क्योंकि उसे execute होने के लिए चाहिए वो उसके पास है।

Q) अगर हमें library file में printf को update करना हो तो क्या करे!
Ans) Printf के source code को change कर update कर देंगे इससे पहले compile हुए code से printf पुराने library के हिसाब से print होगा, अगर हम re-compile करेंगे तो printf change कर देगा change कर देगा यह एक drawback है Compile time linking का

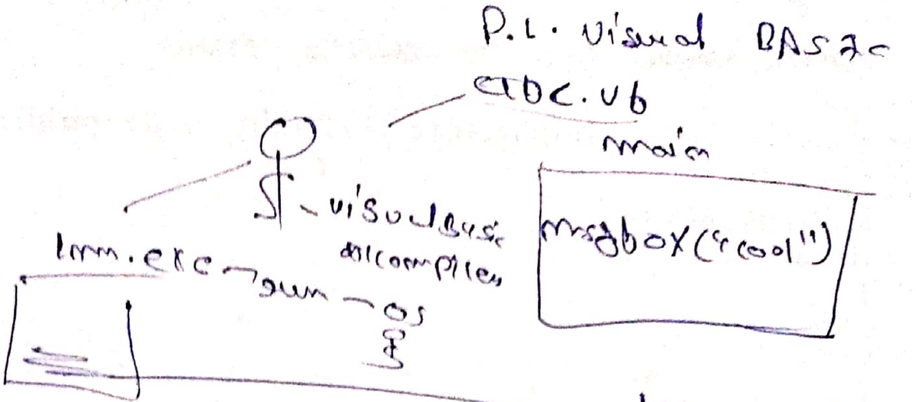
Inheritance (Visibility Mode :public)
C++

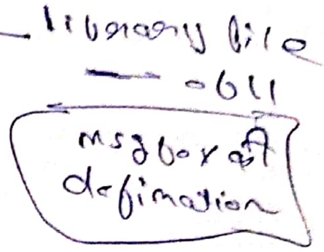
```
#include<iostream.h>
class aaa
{
public:
void sam()
{
cout<<"Hello"<<endl;
}
};
class bbb:public aaa
{
public:
void tom()
{
cout<<"Hi"<<endl;
}
};
void main()
{
bbb *b=new bbb;
b->sam();
b->tom();
}
```

Inheritance (Visibility Mode:private)
C++

```
#include<iostream.h>
class aaa
{
public:
void sam()
{
cout<<"Hello"<<endl;
}
};
class bbb:private aaa
{
public:
void tom()
{
cout<<"Hi"<<endl;
}
};
void main()
{
bbb *b=new bbb;
b->sam();
b->tom();
}
```

इस problem का Solution है Run time linking
के द्वारा

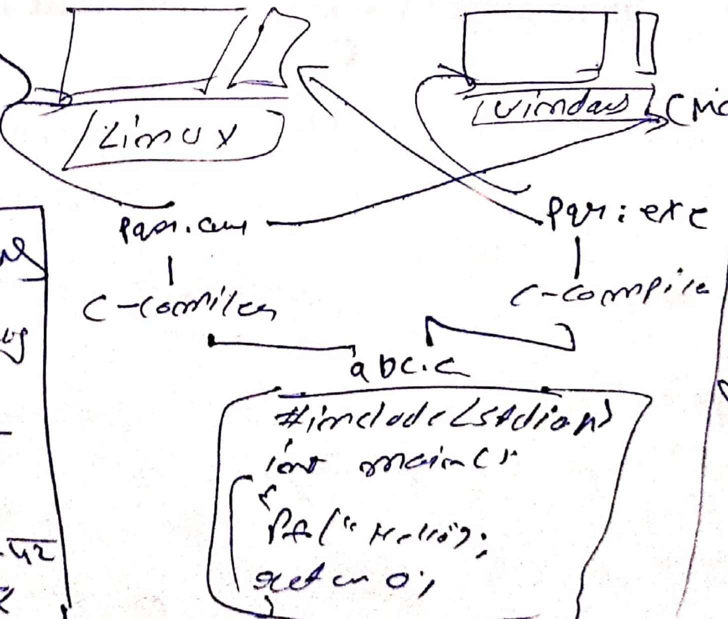




ये lmm.exe Secondary storage में पड़ी होती as इसे RAM में load करता जिसके साथ एक main function की defination होती। अब code execute होगा, तो main की पहली copy बनाएगा और RAM में main की defination load करता फिर msgbox को call करता जिसके msgbox की copy बनाती और library file से defination load कर देता।

→ Java support Run time linking.

→ (mac la)



जिस O.S के लिए code compile हुआ उसी O.S के लिए Run होगा

Linux or windows की machine language different होती है इसलिए C जिस machine पर compile हुआ उसी पर run होगा इसलिए C का platform dependent language कहते है

→ Java is a platform Independent language.
→ किसी भी O.S के लिए code compile हुआ और किसी भी O.S के लिए run हो जाएगा

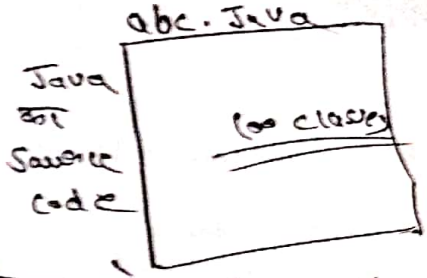
Inheritance
Example 4

```
class Rectangle
{
private int length;
private int breadth;
public void setLength(int length)
{
this.length=length;
}
public void setBreadth(int breadth)
{
this.breadth=breadth;
}
public int getBreadth()
{
return breadth;
}
public int getLength()
{
return length;
}
}
class Box extends Rectangle
{
private int height;
public void setHeight(int height)
{
this.height=height;
}
public int getHeight()
{
return height;
}
}
class eg4psp
{
public static void main(String gg[])
{
Box x=new Box();
x.setLength(10);
x.setBreadth(3);
x.setHeight(4);
System.out.println("Length : "+x.getLength());
System.out.println("Breadth : "+x.getBreadth());
System.out.println("Height : "+x.getHeight());
}
}
```

Length : 10
Breadth : 3
Height : 4

Multiple Inheritance and problems associated with it
C++

```
#include<iostream.h>
```



Javac abc.Java

Compile
⇓
100 class files will be produced

जब abc.Java Compile होगा तो 100 classes files बनेंगे
और Pgm.Java Compile होगा और 20 classes files बनेंगे
और एक Class का file का नाम same हुआ तो वो
override हो जाएगी।

① Java is not support visibility mode.
② Java में Multiple Inheritance नहीं है


```
class aaa
{
public:
void sam()
{
cout<<"Hello"<<endl;
}
};
class bbb:public aaa
{
};
class ccc:public aaa
{
};
class ddd:public bbb,public ccc
{
};
void main()
{
ddd *d;
d=new ddd;
d->sam();
}
```

Virtual Inheritance
C++

Hello

```
#include<iostream.h>
class aaa
{
public:
void sam()
{
cout<<"Hello"<<endl;
}
};
class bbb:virtual public aaa
{
};
class ccc:virtual public aaa
{
};
class ddd:public bbb,public ccc
{
};
void main()
{
ddd *d;
d=new ddd;
d->sam();
}
```

Static Method
Example 5

```
class aaa
{
public static void sam()
{
System.out.println("I have fixed behaviour");
}
public void tom()
{
System.out.println("I change my behaviour");
}
}
class eg5psp
{
public static void main(String gg[])
{
aaa.sam();
aaa.tom();
aaa a=new aaa();
a.sam();
a.tom();
}
}
```

error: non-static method tom() cannot be referenced from a static context
(7: aaa.tom();

Static property
Example 6

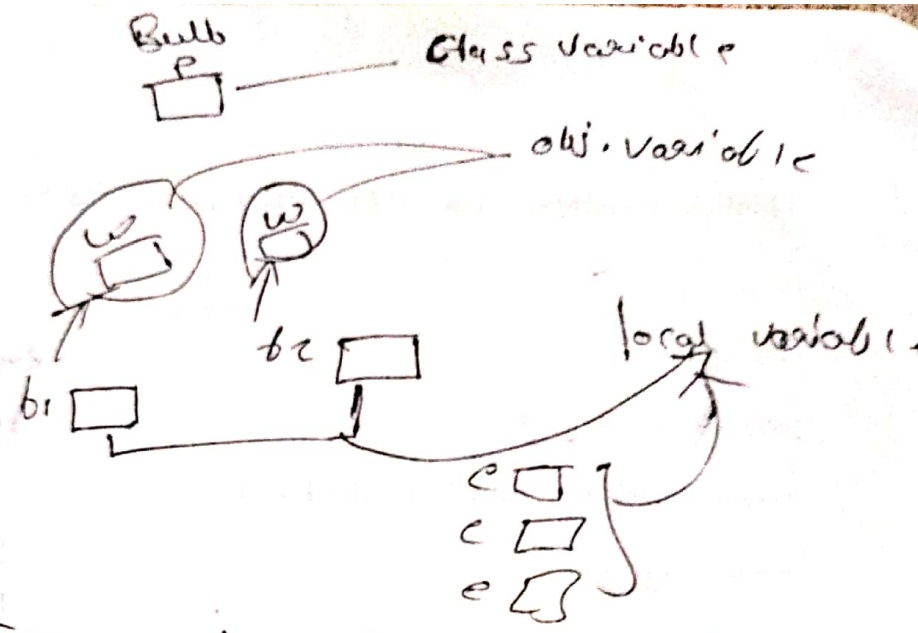
```
class Bulb
{
private int w;
static private int p;
public void setWattage(int e)
{
if(e>=0 && e<=240)
{
w=e;
}
else
{
w=0;
}
}
public int getWattage()
{
return w;
}
public static void setPrice(int e)
{
p=e;
}
public static int getPrice()
{
return p;
}
```

[class name :: static property] C++

C++ में static property को class के बाहर अलग से declare करना पड़ता था जिससे उस static property के लिए memory Allocate हो जाती थी पर Java में class के बाहर declare नहीं करना पड़ता है। उस class का object बनता है तो उसमें static property के लिए जगह नहीं होती है।

Q.) इसमें static property के लिए memory कब और कैसे Allocate होती है?

Ans) इस program में जो class है वो secondary storage में पड़ी रहती है। पहले class eg5psp load होगा main से execution होगा और फिर class Bulb load होगा जिससे JVM class load होते ही class के अन्दर जो static property है उसके लिए memory Allocate कर देगी।



Class के बाद जो memory Allocate होती वो static property के लिए Class Variable कहलाते है

Object से जुड़े लिए memory Allocate होती है उसे obj. variable कहते है

जो function से declear है उसे local variable कहते है

जो function के declared है वह local variable कहते है


```
}
}
class eg6psp
{
public static void main(String gg[])
{
Bulb b1=new Bulb();
Bulb b2=new Bulb();
b1.setWattage(60);
b1.setPrice(10);
System.out.println("Wattage is "+b1.getWattage());
System.out.println("Price is "+b1.getPrice());
b2.setWattage(100);
b2.setPrice(15);
System.out.println("Wattage is "+b2.getWattage());
System.out.println("Price is "+b2.getPrice());
System.out.println("Wattage is "+b1.getWattage());
System.out.println("Price is "+b1.getPrice());
}
}
```

Wattage is : 60
Price is 10
Wattage is 100
Price is 15
Wattage is 60
Price is 15

Garbage value

```
#include<iostream.h>
void main()
{
int x;
cout<<x<<endl;
}
```

① Java is a safe P.C. C++

② Java is a safe programming language.

Rules of assignment in context to a local variable

Example 7

```
class eg7psp
{
public static void main(String gg[])
{
int x;
System.out.println(x);
}
}
```

Variable x might not have been intialized

variable declare
value (Access)

System.out.println(x);

Example 8

```
class eg8psp
{
public static void main(String gg[])
{
int x;
x=10;
System.out.println(x);
}
}
```

O/P = 10

variable declare

value (Access)

Example 9

```
class eg9psp
{
```

```
public static void main(String gg[])
{
int x,y;
y=5;
if(y==5)
{
x=10;
}
System.out.println(x);
}
}
```

Example 10

o/p = 10

```
class eg10psp
{
public static void main(String gg[])
{
int x,y;
y=5;
if(y==5)
{
x=10;
}
else
{
x=20;
}
System.out.println(x);
}
}
```

Local variable as final

Example 11

o/p = 11

```
class eg11psp
{
public static void main(String gg[])
{
int x;
x=10;
x=11;
System.out.println(x);
}
}
```

Example 12

```
class eg12psp
{
public static void main(String gg[])
{
final int x;
x=10;
x=11;
System.out.println(x);
}
```

```
}
```

Example 13 ✗ Error

Cannot assign a value to final variable x

```
class eg13psp
{
public static void main(String gg[])
{
final int x=9;
x=10;
x=11;
System.out.println(x);
}
}
```

Example 14 ✗

Variable x might be assigned in loop x=10;

```
class eg14psp
{
public static void main(String gg[])
{
int y;
final int x;
y=1;
while(y<=1)
{
x=10;
y++;
}
}
}
```

Method overriding
Example 15

o/p ⇒ Bruno
Bhow Bhow
Jump

```
class Dog
{
public void printName()
{
System.out.println("Tommy");
}
public void bark()
{
System.out.println("Bhow Bhow");
}
}
class GermanShepard extends Dog
{
public void jump()
{
System.out.println("Jump Jump");
}
public void printName()
{
System.out.println("Bruno");
}
```

```
}
class eg15psp
{
public static void main(String gg[])
{
GermanShepard gs=new GermanShepard();
gs.printName();
gs.bark();
gs.jump();
}
}
```

---

final method

Example 16

Tommy
Meow
Jump Jump

```
class Dog
{
public void printName()
{
System.out.println("Tommy");
}
public void bark()
{
System.out.println("Bhow Bhow");
}
}
class GermanShepard extends Dog
{
public void jump()
{
System.out.println("Jump Jump");
}
public void bark()
{
System.out.println("Meow Meow");
}
}
class eg16psp
{
public static void main(String gg[])
{
GermanShepard gs=new GermanShepard();
gs.printName();
gs.bark();
gs.jump();
}
}
```

Example 17

Error => bark() in GermanShepard cannot override bark() in Dog
public void bark()

```
class Dog
{
public void printName()
{
System.out.println("Tommy");
}
```

```
final public void bark()
{
System.out.println("Bhow Bhow");
}
}
class GermanShepard extends Dog
{
public void jump()
{
System.out.println("Jump Jump");
}
public void bark()
{
System.out.println("Meow Meow");
}
}
class eg17psp
{
public static void main(String gg[])
{
GermanShepard gs=new GermanShepard();
gs.printName();
gs.bark();
gs.jump();
}
}
```

final class
Example 18

Cannot be inherit from final class

```
final  class aaa
{
}
class bbb extends aaa
{
}
class eg18psp
{
public static void main(String gg[])
{
bbb b;
b=new bbb();
}
}
```

Rules of assignment in context to a Object variable
Example 19

```
class aaa
{
}
class bbb
{
private aaa a;
private long b;
private int c;
```

```
private short d;
private byte e;
private double f;
private float g;
private char h;
private boolean i;
public void sam()
{
System.out.println(a);
System.out.println(b);
System.out.println(c);
System.out.println(d);
System.out.println(e);
System.out.println(f);
System.out.println(g);
System.out.println(h);
System.out.println(i);
}
}
class eg19psp
{
public static void main(String gg[])
{
bbb b=new bbb();
b.sam();
}
}
```

Pointer ka default value null hota hai

null
0
0
0
0
0.0
0.0
false

Rules of assignment in context to a Class variable
Example 20

Could not find or load main class eg20psp

```
class aaa
{
}
class bbb
{
static private aaa a;
static private long b;
static private int c;
static private short d;
static private byte e;
static private double f;
static private float g;
static private char h;
static private boolean i;
public void sam()
{
System.out.println(a);
System.out.println(b);
System.out.println(c);
System.out.println(d);
System.out.println(e);
```

```
System.out.println(f);
System.out.println(g);
System.out.println(h);
System.out.println(i);
}
}
class eg20psp
{
public static void main(String gg[])
{
bbb b=new bbb();
b.sam();
}
}
```

Assigning value to class member

C++ ↑ Properties of the class at the time of declaration

```
#include<iostream.h>
class aaa
{
private:
int x=10;
};
void main()
{
}
```

Example 21

op ⇒ 20

```
class aaa
{
public static int x=20;
}
class eg21psp
{
public static void main(String gg[])
{
System.out.println(aaa.x);
}
}
```

System.out.println
Example 22

```
class aaa
{
public void sam(int e)
{
System.out.println(e);
}
public void sam(char e)
{
System.out.println(e);
}
}
class bbb
```

```
Class PPP
{
  Public void tiger (int a)
  {
    S.o.p(a);
  }
  Public void tiger (float u)
  {
    S.o.p(u);
  }
}
```

```
Class XYZ
{
  Public static int x = false;
  Public static int y = 36;
  Public static PPP Joke = new PPP();
}
```

```
Class PQR
{
  {
    System.out.println(XYZ.y);
    XYZ.Joke.tiger(2.36f);
  }
}
```

Joke → Class
Joke → ~~obj~~ Pointer
tiger → Method

out ~~क्या~~ pointer है?
System class का
Print stream. - data type का out

Joke is an pointer. Joke जिसे Point करता है उसके लिए tiger method call होती है जिसका parameter float type का है

O/P 10
A

```
{
public static aaa a=new aaa();
}
class eg22psp
{
public static void main(String gg[])
{
bbb.a.sam(10);
bbb.a.sam('A');
}
}
```

Constructors
Example 23

1) default constructors
2) Parameterized constructors

```
class Bulb
{
private int w;
Bulb()
{
w=60;
}
Bulb(int e)
{
w=e;
}
public int getWattage()
{
return w;
}
public void setWattage(int e)
{
w=e;
}
}
class eg23psp
{
public static void main(String gg[])
{
Bulb b1=new Bulb(100);
Bulb b2=new Bulb();
System.out.println(b1.getWattage());
System.out.println(b2.getWattage());
}
}
```

O/P 100
60

Example 24

```
class Bulb
{
private int w;
Bulb(int e)
{
w=e;
}
public int getWattage()
```

```
{
return w;
}
public void setWattage(int e)
{
w=e;
}
}
class eg24psp
{
public static void main(String gg[])
{
Bulb b1=new Bulb(100);
Bulb b2=new Bulb();
System.out.println(b1.getWattage());
System.out.println(b2.getWattage());
}
}
```

constructor Bulb in class Bulb cannot be applied to given types;

Example 25

```
class aaa
{
private int x;
aaa(int e)
{
x=e;
}
public int getX()
{
return x;
}
}
class bbb extends aaa
{
}
class eg25psp
{
public static void main(String gg[])
{
bbb b=new bbb();
System.out.println(b.getX());
}
}
```

उपरोक्त कोड class किसी class के Inherit नहीं करती तो obj. class से Inherit कोई और उसक class के Constructor को वहाँ line पर super(); नहीं लिखेगा।

constructor aaa in class aaa cannot be applied to given types; class bbb extends aaa

super keyword to invoke base class constructor

Example 26

```
class aaa
{
private int x;
aaa(int e)
{
x=e;
}
public int getX()
```

```
{
return x;
}
}
class bbb extends aaa
{
bbb()
{
super(10);
}
}
class eg26psp
{
public static void main(String gg[])
{
bbb b=new bbb();
System.out.println(b.getX());
}
}
```

o/p → 10

Example 27

```
class aaa
{
private int x;
aaa(int e)
{
x=e;
}
public int getX()
{
return x;
}
}
class bbb extends aaa
{
bbb()
{
System.out.println("Ujjain");
super(10);
}
}
class eg27psp
{
public static void main(String gg[])
{
bbb b=new bbb();
System.out.println(b.getX());
}
}
```

Constructor aaa in class aaa cannot be applied to given types; {

Call to super must be first statement in constructor super(10) }

super keyword to access base class member
Example 28

```
class aaa
{
public void sam()
{
System.out.println("Hello");
}
public void tom()
{
System.out.println("Good");
}
}
class bbb extends aaa
{
public void sam()
{
sam();
System.out.println("Great");
}
}
class eg28psp
{
public static void main(String gg[])
{
bbb b=new bbb();
b.sam();
}
}
```

Example 29

```
class aaa
{
public void sam()
{
System.out.println("Hello");
}
public void tom()
{
System.out.println("Good");
}
}
class bbb extends aaa
{
public void sam()
{
super.sam();
System.out.println("Great");
super.sam();
}
}
class eg29psp
{
```

```
public static void main(String gg[])
{
bbb b=new bbb();
b.sam();
}
}
```

---

initializer block
Example 30

```
class aaa
{
aaa()
{
System.out.println("Best");
}
{
System.out.println("Good");
}
{
System.out.println("Better");
}
}
class eg30psp
{
public static void main(String gg[])
{
aaa a1,a2;
System.out.println("Ujjain");
a1=new aaa();
System.out.println("Indore");
a2=new aaa();
}
}
```

Ujjain
Good
Better Best
Indore
Good
Better
Best

---

static initializer block
Example 31

```
class aaa
{
static
{
System.out.println("Bad");
}
aaa()
{
System.out.println("Best");
}
{
System.out.println("Good");
}
{
System.out.println("Better");
}
static
```

Static Initializer Block

Ujjain
Bad
Worst
Good
Better
Best
Indore
Good
Better
Best

```
{
System.out.println("Worst");
}
}
class eg31psp
{
public static void main(String gg[])
{
aaa a1,a2;
System.out.println("Ujjain");
a1=new aaa();
System.out.println("Indore");
a2=new aaa();
}
}
```

Object variable as final
Example 32

O/P => 0

(1) यदि एक obj. variable है इसकी value एक default value होगी जो Compile ussignen करेगा

(2) obj. variable final है तो उसे default value नहीं मिलती उसे value Ussign करना पड़ती है

```
class aaa
{
private int x;
public void sam()
{
System.out.println(x);
}
}
class eg32psp
{
public static void main(String gg[])
{
aaa a=new aaa();
a.sam();
}
}
```

Example 33

Variable x not initialized in the default constructor

final private int x;

```
class aaa
{
final private int x;
public void sam()
{
System.out.println(x);
}
}
class eg33psp
{
public static void main(String gg[])
{
aaa a=new aaa();
a.sam();
}
}
```

Example 34

```
class aaa
{
final private int x;
public void sam()
{
x=10;
System.out.println(x);
}
}
class eg34psp
{
public static void main(String gg[])
{
aaa a=new aaa();
a.sam();
}
}
```

अगर obj. final है तो उसे एक बार ही value Assign कर सकते है। Multiple time बार नहीं कर सकते है

- Cannot assign a value to final variable x
  x = 10;

Example 35

O/P => 10

```
class aaa
{
final private int x=10;
public void sam()
{
System.out.println(x);
}
}
class eg35psp
{
public static void main(String gg[])
{
aaa a=new aaa();
a.sam();
}
}
```

Example 36

O/P => 20

```
class aaa
{
final private int x;
public void sam()
{
System.out.println(x);
}
aaa()
{
x=20;
}
}
class eg36psp
{
public static void main(String gg[])
{
```

```
aaa a=new aaa();
a.sam();
}
}
```

Example 37

```
class aaa
{
final private int x=10;
public void sam()
{
System.out.println(x);
}
aaa()
{
x=20;
}
}
class eg37psp
{
public static void main(String gg[])
{
aaa a=new aaa();
a.sam();
}
}
```

Cannot assign a value to final variable x

Example 38

```
class aaa
{
private int y;
final private int x;
public void sam()
{
System.out.println(x);
}
aaa()
{
x=20;
}
aaa(int e)
{
y=e;
}
}
class eg38psp
{
public static void main(String gg[])
{
aaa a=new aaa();
a.sam();
}
}
```

variable x might not have been initialized

Example 39

O/P = 20, 20

```
class aaa
{
private int y;
final private int x;
public void sam()
{
System.out.println(x);
}
aaa()
{
x=20;
}
aaa(int e)
{
y=e;
x=30;
}
}
class eg39psp
{
public static void main(String gg[])
{
aaa a1=new aaa();
a1.sam();
aaa a2=new aaa();
a2.sam();
}
}
```

Example 40

Initilizer block को constructor से पहले run करता है

variable x might already have been assigned x=20;

Variable x ——————

x = 30;

```
class aaa
{
private int y;
final private int x;
{
x=50;
}
public void sam()
{
System.out.println(x);
}
aaa()
{
x=20;
}
aaa(int e)
{
y=e;
x=30;
}
}
class eg40psp
```

```
{
public static void main(String gg[])
{
aaa a1=new aaa();
a1.sam();
aaa a2=new aaa();
a2.sam();
}
}
```

Class variable as final

Example 41

variable x not initialized in the default constructor private final static int x;

```
class aaa
{
private final static int x;
public static void sam()
{
System.out.println(x);
}
}
class eg41psp
{
public static void main(String gg[])
{
aaa.sam();
}
}
```

Example 42

Cannot assign a value to final variable x

```
class aaa
{
private final static int x;
public static void sam()
{
x=10;
System.out.println(x);
}
}
class eg42psp
{
public static void main(String gg[])
{
aaa.sam();
}
}
```

Example 43

o/p => 10

```
class aaa
{
private final static int x=10;
public static void sam()
{
System.out.println(x);
}
```

```
}
class eg43psp
{
public static void main(String gg[])
{
aaa.sam();
}
}
```

Example 44

```
class aaa
{
private final static int x;
static
{
x=10;
}
public static void sam()
{
System.out.println(x);
}
}
class eg44psp
{
public static void main(String gg[])
{
aaa.sam();
}
}
```

Java is a strongly typed language

Example 45

```
class eg45psp
{
public static void main(String gg[])
{
float e;
e=2.33;
float g;
g=2.33f;
}
}
```

Example 46

```
class eg46psp
{
public static void main(String gg[])
{
int x;
float e=2.35;
x=e;
int y;
y=(int)e;
}
```

```
}
```

Example 47

```
class eg47psp
{
public static void main(String gg[])
{
int x;
x=(int)true;
}
}
```

boolean type

Base class reference variable can store reference of a derived class object

Example 48

Error - Incompatible type

bbb cannot be converted to aaa a=new bbb();

```
class aaa
{
}
class bbb
{
}
class eg48psp
{
public static void main(String gg[])
{
aaa a;
a=new bbb();
}
}
```

Example 49

Base class के pointer में Derive class के obj. का address रख सकते है

x

Derive class के pointer में Base class के obj. का address नहीं रख सकते है

```
class aaa
{
}
class bbb extends aaa
{
}
class eg49psp
{
public static void main(String gg[])
{
aaa a;
a=new bbb();
}
}
```

Example 50

Error ⇒ Incompatible types:
aaa cannot be converted to bbb
b=new aaa();

```
class aaa
{
}
class bbb extends aaa
{
}
class eg50psp
{
public static void main(String gg[])
```

```
{
bbb b;
b=new aaa();
}
}
```

Example 51

```
class aaa
{
public void sam()
{
System.out.println("Hello");
}
}
class bbb extends aaa
{
public void tom()
{
System.out.println("good");
}
}
class eg51psp
{
public static void main(String gg[])
{
aaa a;
a=new bbb();
a.sam();
a.tom();
}
}
```

Virtual function
C++

```
#include<iostream.h>
class aaa
{
public:
void sam()
{
cout<<"Hello"<<endl;
}
void tom()
{
}
virtual void john()
{
}
};
class bbb:public aaa
{
public:
void tom()
```

```
{
cout<<"Great"<<endl;
}
void john()
{
cout<<"Good"<<endl;
}
};
void main()
{
aaa *p;
p=new bbb();
p->sam();
p->tom();
p->john();
}
```

Virtual polymorphism
Example 52

O/P => Hello
Good

```
class aaa
{
public void sam()
{
System.out.println("Hello");
}
public void tom()
{
}
}
class bbb extends aaa
{
public void tom()
{
System.out.println("good");
}
}
class eg52psp
{
public static void main(String gg[])
{
aaa a;
a=new bbb();
a.sam();
a.tom();
}
}
```

abstract method and abstract class
Example 53

```
class aaa
{
public void sam()
}
```

Example 54

```
class aaa
{
public void sam();
}
```

Example 55

```
class aaa
{
abstract public void sam();
}
```

Example 56

```
abstract class aaa
{
abstract public void sam();
}
```

Example 57

```
abstract class aaa
{
abstract public void sam();
public void tom()
{
System.out.println("God is great");
}
}
class eg57psp
{
public static void main(String g[])
{
aaa a;
}
}
```

Example 58

```
abstract class aaa
{
abstract public void sam();
}
class eg58psp
{
public static void main(String g[])
{
aaa a;
a=new aaa();
}
}
```

Example 59

```
abstract class aaa
{
abstract public void sam();
}
class bbb extends aaa
{
```

```
public void lion()
{
System.out.println("cool");
}
}
```

Example 60

```
abstract class aaa
{
abstract public void sam();
}
abstract class bbb extends aaa
{
public void lion()
{
System.out.println("cool");
}
}
```

Example 61

```
abstract class aaa
{
abstract public void sam();
}
class bbb extends aaa
{
public void lion()
{
System.out.println("cool");
}
public void sam()
{
System.out.println("good");
}
}
```

Promotion
Example 62

```
class aaa
{
public void sam(long g)
{
System.out.println("long : "+g);
}
public void sam(int g)
{
System.out.println("int : "+g);
}
}
class eg62psp
{
public static void main(String g[])
{
aaa a=new aaa();
```

```
long e=20;
int f=20;
a.sam(e);
a.sam(f);
}
}
```

Example 63

```
class aaa
{
public void sam(long g)
{
System.out.println("long : "+g);
}
}
class eg63psp
{
public static void main(String g[])
{
aaa a=new aaa();
long e=20;
int f=20;
a.sam(e);
a.sam(f);
}
}
```

Example 64

```
class aaa
{
public void sam(int g)
{
System.out.println("int : "+g);
}
}
class eg64psp
{
public static void main(String g[])
{
aaa a=new aaa();
long e=20;
int f=20;
a.sam(e);
a.sam(f);
}
}
```

Example 65

```
class aaa
{
public void sam(int g)
{
System.out.println("int : "+g);
}
```

o/p = 201
20

```
}
class eg65psp
{
public static void main(String g[])
{
aaa a=new aaa();
long e=20;
int f=20;
a.sam((int)e); → type casting करदी long को int में
a.sam(f);
}
}
```

Donkey – Monkey Wala Example
Example 66

```
abstract class Animal
{
public int getAge()
{
return 10;
}
}
class Donkey extends Animal
{
public int getAge()
{
return 15;
}
}
class Monkey extends Animal
{
public int getage()
{
return 20;
}
}
class Tommy
{
public int getAge()
{
return 50;
}
}
class Lion
{
public void eat(Animal a)
{
int x;
x=a.getAge();
System.out.println(x);
}
}
class Zoo
```

```
{
public static Lion sherKhan=new Lion();
}
class eg66psp
{
public static void main(String gg[])
{
Donkey d=new Donkey();
Monkey m=new Monkey();
Tommy t=new Tommy();
Zoo.sherKhan.eat(d);
Zoo.sherKhan.eat(m);
Zoo.sherKhan.eat(t);
}
}
```

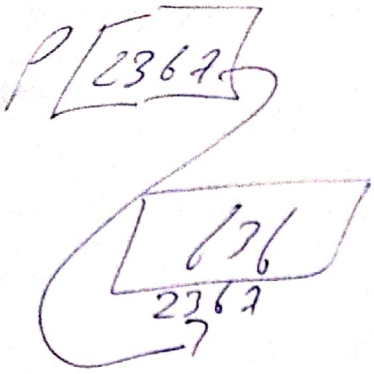
Object is the super most class in java

Example 67

```
class aaa
{
public void sam()
{
System.out.println("God is great");
}
}
class eg67psp
{
public static void main(String gg[])
{
Object j;
j=new aaa();
}
}
```

Example 68

```
class aaa
{
public void sam()
{
System.out.println("God is great");
}
}
class eg68psp
{
public static void main(String gg[])
{
Object j;
j=new aaa();
j.sam();
}
}
```




```
int *p;
p = (int *) 2367;
*p = 636;
```

```
int x[5];
x[200] = 60;
```

```
x[-30] = 735;
```

```
int x[5];
*(x+2) = 30;
```

x[2]

Special Treatment

1) Array
2) String
3) Boxing / unboxing
3) Class loading
4) lambda
5) Anonymous

~~1. Scanclass sc = ...~~

Unsafe code
C++

```
#include<iostream.h>
class aaa
{
public:
void sam()
{
cout<<"Hello"<<endl;
}
};
void main()
{
int *x;
x=(int *)500;
*x=50;
int y[5];
y[-3]=200;
y[100]=399;
x=y;
*(x+500)=300;
aaa *a;
a=(aaa *)500;
*a=3432;
}
```

Array is treated as an object
Example 69

```
class eg69psp
{
public static void main(String gg[])
{
int x[];
x=new int[5];
x[0]=30;
x[1]=43;
x[2]=44;
x[4]=39;
int e;
for(e=0;e<x.length;e++)
{
System.out.println(x[e]);
}
}
}
```

Example 70

```
class eg70psp
{
public static void main(String gg[])
{
int x[];
x=new int[3];
```

Release

Java में memory release नहीं होती है। अगर program में कोई obj. ऐसी जो किसी को भी point नहीं कर रहा तो वह memory automatically release हो जाएगी।

C++

```
int *x;
x = new int;
->S.O.P(x);
delet x;
-> S.O.P(x);
*x = 600;
```

```
x[0]=30;
x[1]=43;
x[2]=44;
System.out.println("Ujjain");
x[3]=54;
System.out.println("Indore");
x[4]=39;
System.out.println("Goa");
}
}
```

Ujjain
Index 3 out of Bounds for length 3

Resizing array
Example 71

```
class eg71psp
{
public static void main(String gg[])
{
int [] x;
x=new int[3];
x[0]=30;
x[1]=43;
x[2]=44;
int [] t;
t=new int[5];
int e;
for(e=0;e<x.length;e++)
{
t[e]=x[e];
}
x=t;
x[3]=54;
x[4]=39;
for(e=0;e<x.length;e++)
{
System.out.println(x[e]);
}
}
}
```

O/P =>
30
43
44
54
39

Array of reference variables
Example 72

```
class Bulb
{
private int w;
public void setWattage(int e)
{
w=e;
}
public int getWattage()
{
return w;
}
}
```

Ujjain

Exception in thread "main" java.lang.NullPointer
Exception at eg72PSP.m

```
class eg72psp
{
public static void main(String gg[])
{
Bulb b[];
b=new Bulb[2];
System.out.println("Ujjain");
b[0].setWattage(60);
System.out.println("Indore");
b[1].setWattage(100);
System.out.println("Goa");
System.out.println(b[0].getWattage());
System.out.println(b[1].getWattage());
}
}
```

Example 73

```
class Bulb
{
private int w;
public void setWattage(int e)
{
w=e;
}
public int getWattage()
{
return w;
}
}
class eg73psp
{
public static void main(String gg[])
{
Bulb b[];
b=new Bulb[2];
b[0]=new Bulb();
b[1]=new Bulb();
b[0].setWattage(60);
b[1].setWattage(100);
System.out.println(b[0].getWattage());
System.out.println(b[1].getWattage());
}
}
```

String
Example 74

```
class aaa
{
}
class eg74psp
{
public static void main(String gg[])
{
aaa a;
```

```
a="Hello";
}
}
```

Example 75

O/P => Same
Not Same

```
class eg75psp
{
public static void main(String gg[])
{
String g;
g="Hello";
String t;
t="Good";
String k;
k="Hello";
if(g==k)
{
System.out.println("Same");
}
else
{
System.out.println("Not Same");
}
String r;
r=new String("Good");
if(t==r)
{
System.out.println("Same");
}
else
{
System.out.println("Not Same");
}
}
}
```

String को Data (V) में जो assign करते हैं वो String मानता है

Special treatment compiler को यह सिखाता है कि जब भी String class के pointer को "" का प्रयोग करते हुए assign करें तो String मानना है

JVM main a (D.S) Pool of String type OS.

println with Object as parameter and toString method
Example 76

Great

```
/*
// strictly not for practical
class PrintStream
{
public void println(long e)
{
// some code to print the value of e
}
public void println(int e)
{
// some code to print the value of e
}
public void println(double e)
{
// some code to print the value of e
}
```

```
public void println(float e)
{
// some code to print the value of e
}
public void println(char e)
{
// some code to print the value of e
}
public void println(boolean e)
{
// some code to print the value of e
}
public void println(Object e)
{
// through (e) toString() method gets called and
// whatever toString returns, it gets printed
}
}
// Strictly not for practicals
class System
{
final public static PrintStream out=new PrintStream();
}
*/
class aaa
{
public String toString()
{
return "Great";
}
}
class eg76psp
{
public static void main(String gg[])
{
aaa a;
a=new aaa();
System.out.println(a);
}
}
```

Example 77

aaa@4517 d9 a3

```
class aaa
{
}
class eg77psp
{
public static void main(String gg[])
{
aaa a;
a=new aaa();
System.out.println(a);
}
```

}

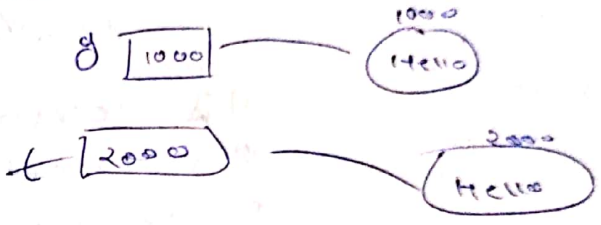
Example 78

O/P ⇒ Hello

```
class eg78psp
{
public static void main(String gg[])
{
String g;
g="Hello";
System.out.println(g);
}
}
```

Deep comparison
Example 79

O/P ⇒ Same



g के लिए memory Allocate होगी
एक special treatment दी
provide होगा और एक obj.
Create होगा जिसका address
g को assign कर दिया जाएगा
और Hello वहां रख देगा.

```
class eg79psp
{
public static void main(String gg[])
{
String g="Hello";
String t=new String("Hello");
if(g.equals(t)==true)
{
System.out.println("Same");
}
else
{
System.out.println("Not Same");
}
}
}
```

lexical comparison
Example 80

O/P ⇒ -1
1
0

```
class eg80psp
{
public static void main(String gg[])
{
String a="AMIT";
String b="BOBBY";
String c="ANURAG";
String d="ANURAG";
System.out.println(a.compareTo(b));
System.out.println(b.compareTo(c));
System.out.println(c.compareTo(d));
}
}
```

Exception handling
Example 81

Exception in thread "main" java.lang.Arithmetic Exception: / by zero at eg81PSP.main

```
class eg81psp
{
public static void main(String gg[])
{
int x,y,z;
```

— to upper • compare to (toUpper, compare To)

- In Java string are Imutable it means we cannot change in Java

Object

Throwable

Exception — Error

1) Io Exception
2) So Exception
3) Interrupted Exception
4) Remove exception
5) Parse exception
6) Class Not found exception.
7) Runtime exception
   - → Arithmetic exception
   - → Number format exception
   - → Array Index out of Bound exception.

Exception → ये problem code से generate ~~होते है~~ है होते

error ⇒ ये problem storage purpose से या O.S से होते है यह problem को हम error नहीं कर सकते। क्योंकि यह code ~~other system~~ ये है problem होते है तो हम इसे exception से solve करेंगे।

```
x=10;
y=0;
z=x/y;
System.out.println(z);
System.out.println("Neat End");
}
}
```

Example 82 O/P => Neat End

```
class eg82psp
{
public static void main(String gg[])
{
int x,y,z;
x=10;
y=0;
try
{
z=x/y;
System.out.println(z);
}catch(ArithmeticException ae)
{
}
System.out.println("Neat End");
}
}
```

Example 83 y reset to 1 to Neat End

```
class eg83psp
{
public static void main(String gg[])
{
int x,y,z;
x=10;
y=0;
try
{
z=x/y;
System.out.println(z);
}catch(ArithmeticException ae)
{
System.out.println("y reset to 1");
y=1;
z=x;
System.out.println(z);
}
System.out.println("Neat End");
}
}
```

Pointer

of How JVM runs the code?

Java PSP

① will load PSP class
② will place a call to main (entry point function)

try
{

}catch(Throwable t)
{
t.printStackTrace();
System.exit(0);
}

App. ends

Catch is not a function.
Catch is a Pointer.
data type of Catch is Throwable,
Throwable is not Parameter.

z = x/y — Exception is raised.

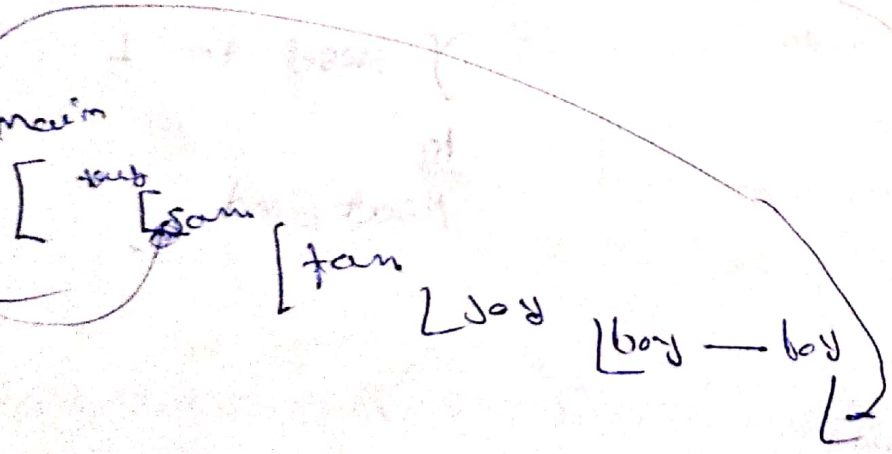
JVM handle the exception handling.

z = x/y

Behind the Screen

Arithmetic Exception ae
ae = Arithmetic ("Divide by zero");
throw ae;
1000

Exception handling is not about



try with multiple catch blocks
Example 84

```
class eg84psp
{
public static void main(String gg[])
{
int j[];
j=new int[5];
int x,y,z;
x=20;
y=2;
try
{
z=x/y;
System.out.println(z);
j[z]=33;
System.out.println(j[z]);
}catch(ArithmeticException ae)
{
System.out.println("Caught AE");
}
catch(ArrayIndexOutOfBoundsException xe)
{
System.out.println("Caught AIOOBE");
}
System.out.println("Neat End");
}
}
```

10
Caught A IooB
Neat End

Example 85

```
class aaa
{
public void sam()
{
int x,y,z;
x=10;
y=0;
z=x/y;
System.out.println(z);
}
}
class eg85psp
{
public static void main(String g[])
{
aaa a=new aaa();
try
{
a.sam();
}catch(ArithmeticException ae)
{
System.out.println("Caught AE");
}
```

Caught AE
Neat Edit

```
System.out.println("Neat Edit");
}
}
```

nested try - catch
Example 86

O/P > Caught AE

```
class eg86psp
{
public static void main(String gg[])
{
int x,y,z;
x=10;
y=0;
int j[];
j=new int[3];
try
{
z=x/y;
System.out.println(z);
try
{
j[10]=34;
System.out.println(j[10]);
}catch(ArrayIndexOutOfBoundsException xe)
{
System.out.println("Caught AIOOBE");
}
}catch(ArithmeticException xe)
{
System.out.println("Caught AE");
}
}
}
```

Example 87

5
Caught AIooBE
Onest

```
class eg87psp
{
public static void main(String gg[])
{
int x,y,z;
x=10;
y=2;
int j[];
j=new int[3];
try
{
z=x/y;
System.out.println(z);
try
{
j[10]=34;
System.out.println(j[10]);
}catch(ArrayIndexOutOfBoundsException xe)
{
```

```
System.out.println("Caught AIOOBE");
}
System.out.println("Great");
}catch(ArithmeticException xe)
{
System.out.println("Caught AE");
}
}
}
```

Example 88

o/p ⇒ 5
Caught AE

```
class eg88psp
{
public static void main(String gg[])
{
int x,y,z;
x=10;
y=2;
int j[];
j=new int[3];
try
{
z=x/y;
System.out.println(z);
try
{
x=10;
y=0;
z=x/y;
System.out.println(z);
j[10]=34;
System.out.println(j[10]);
}catch(ArrayIndexOutOfBoundsException xe)
{
System.out.println("Caught AIOOBE");
}
System.out.println("Great");
}catch(ArithmeticException xe)
{
System.out.println("Caught AE");
}
}
}
```

Example 89

Exception in thread "main" java. ~~lang.Array~~ lang.IndexOutOfBoundsExc
Index 40 out of bound s for length 3

```
class eg89psp
{
public static void main(String gg[])
{
int x,y,z;
x=10;
y=2;
int j[];
j=new int[3];
```

```
try
{
j[40]=30;
z=x/y;
System.out.println(z);
try
{
System.out.println(z);
j[10]=34;
System.out.println(j[10]);
}catch(ArrayIndexOutOfBoundsException xe)
{
System.out.println("Caught AIOOBE");
}
System.out.println("Great");
}catch(ArithmeticException xe)
{
System.out.println("Caught AE");
}
}
}
```

catch with super class reference variable

Example 90

```
class eg90psp
{
public static void main(String gg[])
{
try
{
// some 50 lines of code
}catch(ArithmeticException ae)
{

}
catch(ArrayIndexOutOfBoundsException xe)
{

}
catch(Exception e)
{
}
}
}
```

Example 91

→ Exception already ArrayIndexOutOfBoundsException has already been caught
catch (Array Index — xe
→ ——————
catch (Array — e)

```
class eg91psp
{
public static void main(String gg[])
{
try
{
// some 50 lines of code
```

```
}catch(Exception ae)
{

}
catch(ArrayIndexOutOfBoundsException xe)
{

}
catch(ArithmeticException e)
{
}
}
}
```

Caught AE

Creating and throwing custom exceptions

Example 92

```
class FinanceCalculator
{
public int calculateEligibility(int familyMembers,int income)
{
int loanAmount;
loanAmount=(income/familyMembers)*15;
return loanAmount;
}
}
class eg92psp
{
public static void main(String gg[])
{
int familyMembers,income,loanAmount;
income=15000;
familyMembers=0;
FinanceCalculator fc=new FinanceCalculator();
try
{
loanAmount=fc.calculateEligibility(familyMembers,income);
System.out.println("Loan Amount "+loanAmount);
}catch(ArithmeticException ae)
{
System.out.println("Caught AE");
// some code to rectify the exception

}
}
}
```

OPS ⇒ Loan Amount -75000

Example 93

```
class FinanceCalculator
{
public int calculateEligibility(int familyMembers,int income)
{
int loanAmount;
loanAmount=(income/familyMembers)*15;
return loanAmount;
```

```
}
}
class eg93psp
{
public static void main(String gg[])
{
int familyMembers,income,loanAmount;
income=15000;
familyMembers=-3;
FinanceCalculator fc=new FinanceCalculator();
try
{
loanAmount=fc.calculateEligibility(familyMembers,income);
System.out.println("Loan Amount "+loanAmount);
}catch(ArithmeticException ae)
{
System.out.println("Caught AE");
// some code to rectify the exception

}
}
}
```

Example 94

```
class FinanceCalculator
{
public int calculateEligibility(int familyMembers,int income)
{
if(familyMembers<0)
{
return -1;
}
int loanAmount;
loanAmount=(income/familyMembers)*15;
return loanAmount;
}
}
class eg94psp
{
public static void main(String gg[])
{
int familyMembers,income,loanAmount;
income=15000;
familyMembers=-3;
FinanceCalculator fc=new FinanceCalculator();
try
{
loanAmount=fc.calculateEligibility(familyMembers,income);
if(loanAmount==-1)
{
System.out.println("Family members cannot be negative");
}
else
```

```
{
System.out.println("Loan Amount "+loanAmount);
}
}catch(ArithmeticException ae)
{
System.out.println("Caught AE");
// some code to rectify the exception

}
}
}
```

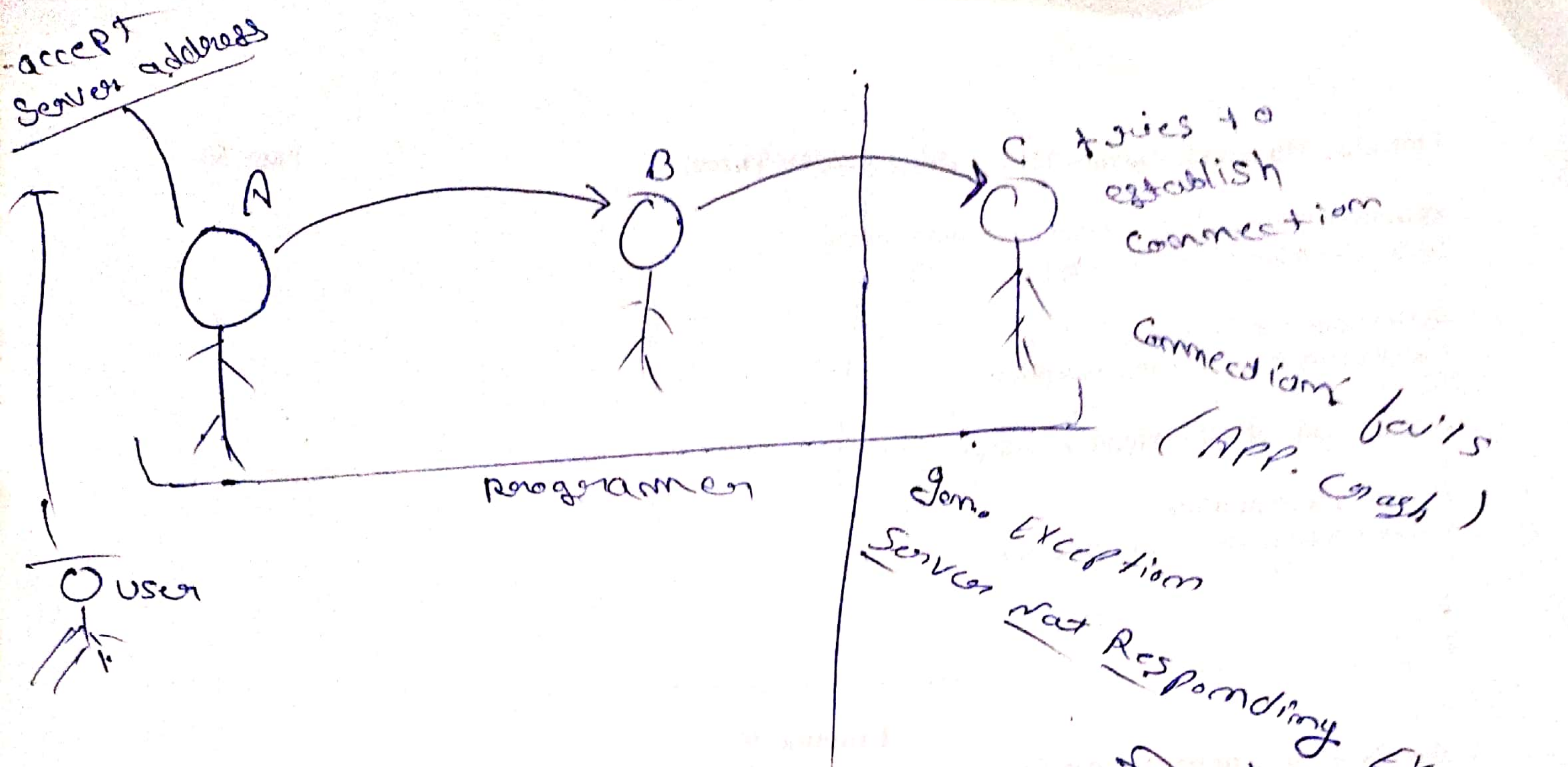
Example 95

```
class NegativeFamilyMembersException
{
private String message;
NegativeFamilyMembersException(String message)
{
this.message=message;
}
public String getMessage()
{
return message;
}
public String toString()
{
return message;
}
}
class FinanceCalculator
{
public int calculateEligibility(int familyMembers,int income)
{
if(familyMembers<0)
{
return -1;
}
int loanAmount;
loanAmount=(income/familyMembers)*15;
return loanAmount;
}
}
class eg95psp
{
public static void main(String gg[])
{
int familyMembers,income,loanAmount;
income=15000;
familyMembers=-3;
FinanceCalculator fc=new FinanceCalculator();
try
{
loanAmount=fc.calculateEligibility(familyMembers,income);
```

```
System.out.println("Loan Amount "+loanAmount);
}catch(ArithmeticException ae)
{
System.out.println("Caught AE");
// some code to rectify the exception
}
catch(NegativeFamilyMembersException nfe)
{
System.out.println(nfe);
// some code to rectify the exception
}
}
}
```

Example 96

```
class NegativeFamilyMembersException extends RuntimeException
{
private String message;
NegativeFamilyMembersException(String message)
{
this.message=message;
}
public String getMessage()
{
return message;
}
public String toString()
{
return message;
}
}
class FinanceCalculator
{
public int calculateEligibility(int familyMembers,int income)
{
if(familyMembers<0)
{
NegativeFamilyMembersException n;
n=new NegativeFamilyMembersException("Family Members Cannot Be Negative");
throw n;
}
int loanAmount;
loanAmount=(income/familyMembers)*15;
return loanAmount;
}
}
class eg96psp
{
public static void main(String gg[])
{
int familyMembers,income,loanAmount;
income=15000;
```



⇒ Connection fail गलत Server address के वजह से हुआ ऐसा हो तो एक Exception बनाना पड़ेगा generate करना पड़ेगा जो गलत add. को सही करे।

print()

```
Class ServerNotResponding extends Exception
{
  SerNotRes ( Stri. s )
  { super(s);
  }
}
```

```
familyMembers=-3;
FinanceCalculator fc=new FinanceCalculator();
try
{
loanAmount=fc.calculateEligibility(familyMembers,income);
System.out.println("Loan Amount "+loanAmount);
}catch(ArithmeticException ae)
{
System.out.println("Caught AE");
// some code to rectify the exception
}
catch(NegativeFamilyMembersException nfe)
{
System.out.println("Caught NFME");
System.out.println(nfe);
// some code to rectify the exception
}
}
}
```

Checked-unchecked Exception

Caught – Uncaught Exceptions and throws keyword

Example 97

Exception in thread 'main' aaa
at bbb.sam
at eg97psp.main

```
class aaa extends RuntimeException
{
}
class bbb
{
public void sam()
{
// some code to identify problem
aaa a;
a=new aaa();
throw a;
}
}
class eg97psp
{
public static void main(String gg[])
{
bbb b;
b=new bbb();
b.sam();
}
}
```

Example 98

Unchecked exception can
must be caught or
declared to be thrown
throw a;

```
class aaa extends Exception
{
}
class bbb
{
public void sam()
{
// some code to identify problem
```

```
aaa a;
a=new aaa();
throw a;
}
}
class eg98psp
{
public static void main(String gg[])
{
bbb b;
b=new bbb();
b.sam();
}
}
```

Example 99

Caught aaa

```
class aaa extends Exception
{
}
class bbb
{
public void sam()
{
// some code to identify problem
aaa a;
a=new aaa();
try
{
throw a;
}catch(aaa x)
{
System.out.println("Caught aaa");
}
}
}
class ccc
{
public void tom()
{
bbb b;
b=new bbb();
b.sam();
}
}
class eg99psp
{
public static void main(String gg[])
{
ccc c;
c=new ccc();
c.tom();
}
}
```

Example 100

Caught aaa

```
class aaa extends Exception
{
}
class bbb
{
public void sam() throws aaa
{
// some code to identify problem
aaa a;
a=new aaa();
throw a;
}
}
class ccc
{
public void tom()
{
bbb b;
try
{
b=new bbb();
b.sam();
}catch(aaa a)
{
System.out.println("Caught aaa");
}
}
}
class eg100psp
{
public static void main(String gg[])
{
ccc c;
c=new ccc();
c.tom();
}
}
```

Example 101

```
class aaa extends Exception
{
}
class bbb
{
public void sam() throws aaa
{
// some code to identify problem
aaa a;
a=new aaa();
throw a;
}
}
```

Caught aaa

```
class ccc
{
public void tom() throws aaa
{
bbb b;
b=new bbb();
b.sam();
}
}
class eg101psp
{
public static void main(String gg[])
{
ccc c;
c=new ccc();
try
{
c.tom();
}catch(aaa a)
{
System.out.println("Caught aaa");
}
}
}
```

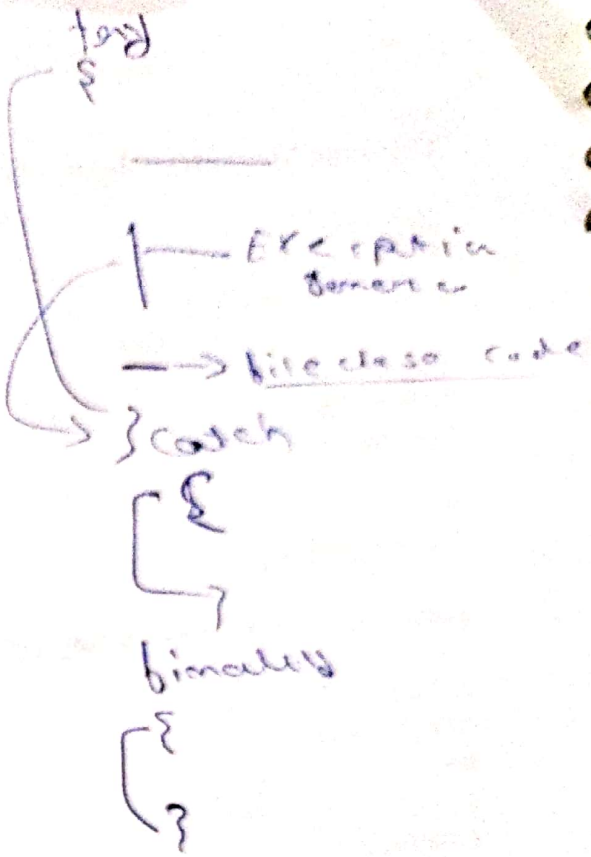
Example 102

Exception in thread "main"

at bbb.Sam
at ccc.tom

```
class aaa extends Exception
{
}
class bbb
{
public void sam() throws aaa
{
// some code to identify problem
aaa a;
a=new aaa();
throw a;
}
}
class ccc
{
public void tom() throws aaa
{
bbb b;
b=new bbb();
b.sam();
}
}
class eg102psp
{
public static void main(String gg[]) throws aaa
{
ccc c;
```

अगर try Block में कोई Exception आता है तो file close नहीं होगी और try Block को and में catch Block में जाएगा और फिर finally Block में जाएगा फिर उसके file close करने का code होगा अगर exception का object उस catch Block से नहीं मिलता है तो finally Block में जाएगा।



```
c=new ccc();
c.tom();
}
}
```

try – catch - finally
Example 103

Caught AE
Finally executed

try Block must be followed by catch block / finally Block followed by finally Block

```
class eg103psp
{
public static void main(String gg[])
{
try
{
int x,y,z;
x=10;
y=0;
z=x/y;
System.out.println(z);
}catch(ArithmeticException ae)
{
System.out.println("Caught AE");
}
finally
{
System.out.println("Finally exexuted");
}
}
}
```

Example 104

5
Finally execu

```
class eg104psp
{
public static void main(String gg[])
{
try
{
int x,y,z;
x=10;
y=2;
z=x/y;
System.out.println(z);
}catch(ArithmeticException ae)
{
System.out.println("Caught AE");
}
finally
{
System.out.println("Finally exexuted");
}
}
}
```

Example 105

```
class eg105psp
{
public static void main(String gg[])
{
try
{
int x,y,z;
x=10;
y=0;
z=x/y;
System.out.println(z);
}
finally
{
System.out.println("Finally exexuted");
}
}
}
```

Finally executed
Exception in thread "main" java.lang.ArithmeticException: / by zero

Parsing String to int
Example 106

```
class Converter
{
public static int convertToInt(String data)
{
int x=0;
int y=1;
int z=data.length()-1;
char g;
while(z>=0)
{
g=data.charAt(z);
if(g>=48 && g<=57)
{
x=x+((g-48)*y);
}
else
{
NumberFormatException nfe=new NumberFormatException();
throw nfe;
}
y=y*10;
z--;
}
return x;
}
}
class eg106psp
{
public static void main(String gg[])
{
String a,b;
```

Value of x is 101

java.lang.NumberFormatException

```
a="101";
b="101good";
int x,y;
try
{
x=Converter.convertToInt(a);
System.out.println("Value of x is "+x);
y=Converter.convertToInt(b);
System.out.println("Value of y is "+y);
}catch(NumberFormatException nfe)
{
System.out.println(nfe);
}
}
}
```

Integer.praseInt
Example 107

Value of x is 101
Java.lang.NumberFormat
For input String: "101g

```
class eg107psp
{
public static void main(String gg[])
{
String a,b;
a="101";
b="101good";
int x,y;
try
{
x=Integer.parseInt(a);
System.out.println("Value of x is "+x);
y=Integer.parseInt(b);
System.out.println("Value of y is "+y);
}catch(NumberFormatException nfe)
{
System.out.println(nfe);
}
}
}
```

Command line arguments
Example 108

O/P => 0

यहाँ String class के obj. बनेगा जिसमे collection of character रखे जायेगा जिसका address string class के obj. को using करेगा। जिसमे यह Array मे रखे होंगे और 2 इस obj. का address main function को दिया जाएगा।

```
class eg108psp
{
public static void main(String gg[])
{
System.out.println(gg.length);
int x;
for(x=0;x<gg.length;x++)
{
System.out.println(gg[x]);
}
}
}
// execution guidelines
```

```
// java eg108psp
// java eg108psp god is great
// java eg108psp ujjain is the "city of gods"
```

Example 109

Usage : java eg109psp num1 num2

```
class eg109psp
{
public static void main(String gg[])
{
if(gg.length!=2)
{
System.out.println("Usage : java eg109psp num1 num2");
}
else
{
try
{
int sum=Integer.parseInt(gg[0])+Integer.parseInt(gg[1]);
System.out.println("Sum "+sum);
}catch(NumberFormatException nfe)
{
System.out.println(nfe);
}
}
}
}
```

Introduction to factory class
Example 110

```
class Bulb
{
private int w;
public void setWattage(int e)
{
w=e;
}
public int getWattage()
{
return w;
}
}
class BulbFactory
{
private BulbFactory()
{
}
public static Bulb prepareBulb(int wattage)
{
Bulb b=null;
if(wattage>=0 && wattage<=240)
{
b=new Bulb();
b.setWattage(60);
}
```

```
return b;
}
}
class eg110psp
{
public static void main(String gg[])
{
Bulb k;
k=BulbFactory.prepareBulb(60);
if(k!=null)
{
System.out.println(k.getWattage());
}
}
}
```