

- $$\frac{1}{2}$$

70

P

- * *

1

1

1

Scanner sc = new Scanner(System.in)

name of class obj name constructor (same as name of class)

A class is a blueprint from which individual objects are created

- > Specifying a class doesn't create any objects of that class
(In the same way specifying a structure in C doesn't create any variable)
- > To create objects we need keyword new
- > The object variable type must be same as class

Creating an object is called instantiating it. **INSTANTIATING**

Object is an instance of a class

The new operator instantiates a class by allocating memory for new object & returning a reference to that memory
also invokes object loader

Methods in JAVA is like functions in C

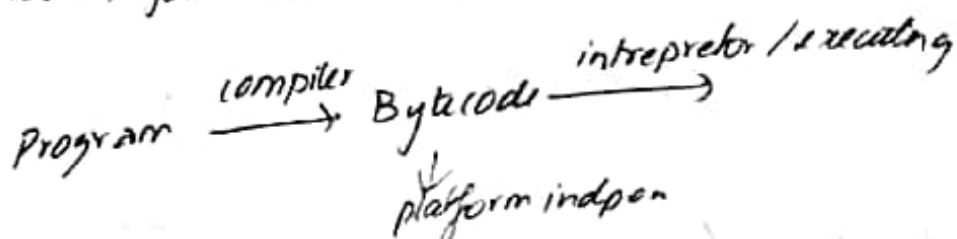
- * The programme starts and runs the main method
- * Anything outside the main method should be called first.

JVM - is an additional layer between the language & OS which makes it platform independent.

static - With this keyword we need not instantiate main
Main is the only method without instantiating the object

void -> doesn't return any argument

String args[] -> for command line prompt



NO IDE can go through command line.

How to debug code using break points

Segmentation fault error

Java Constructor: A constructor in java is a method used to initialise objects

- > constructor has same name as class
- > invoked when obj of class is created
directly creates and initialises.

Try Catch In Java

The try block contains set of statements where an exception can occur. A try block is always followed by a catch block

```
try {  
    // statements that may cause a exception  
}  
catch (Exception type)  
{  
}
```

QUIZ 1 24/6/18

WHAT IS ENCAPSULATION

wrapping data & code to secure them from outside interface

HOW INHERITANCE IS USEFUL FOR PROGRAMMER

Allows only authorised users through class & objects
Creating methods & data from some other class super class

POLYMORPHISM SUPPORTED IN C++ True

HOW MAIN FUNCTION OF JAVA CLASS IS GETTING EXECUTED WITHOUT INSTANTIATION

Only function which runs without instantiating an object
automatically by the compiler

OBJECTS AS PARAMETERS

```
class Box
```

```
{  
    double d, w, h;
```

```
    Box(Box ob)
```

```
    {  
        d = ob.d;
```

```
        w = ob.w;
```

```
        h = ob.h;
```

```
    }
```

```
    Box(double a, double b, double c)
```

```
    {
```

```
        d = a;
```

```
        w = b;
```

```
        h = c;
```

```
    }
```

```
}
```

```
class new
```

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

```
    {  
        Box mybox = new Box(1.1, 2.2, 3.3);
```

```
        Box myclone = new Box(mybox);
```

ACCESS SPECIFIERS

3 types

- public - can be accessed by anybody
- private - only by public methods inside the class
- protected - only in derived class

An access specifier is a defining code element that can determine which elements of a program are allowed to access a specific variable or other piece of data.

Constructor Overloading

class Box

{ double w, d, h;

Box(double a, double b, double c)

{ w = a;

d = b;

h = c; }

Box()

{ w = -1; d = -1; h = -1; }

Box(double len)

{ w = d = h = len; }

double vol()

{ return w * d * h; }

}

- > Objects as arguments
- > Objects as parameters

Method overloading is a feature that allows a class to have more than one method having the same name

- 1) number of parameters `add(int int)` `add(int int int)`
- 2) data type of parameters `add(int, int)` `add(int float)`
- 3) sequence of data type `add(int, float)` `add(float, int)`
- 4) return type `static` or `void`

Constructor overloading

- > Having more than one constructor with different parameters such that each constructor performs a different task

INTERFACES

- > Interface looks like a class but it isn't
- > It can have methods & variables just like class but methods declared as abstract
- > used for full abstraction
- > we can have more than one interface in class

Constructors:

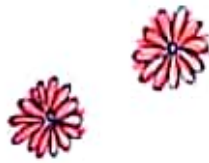
- > A constructor is a method name of constructor is name of class
- > When it created constructor is automatically called
- > no return value
- > It is a block of code similar to other methods but it is called when an instance of that class is created
↓
object

Initialization is the process of creating a class

default constructor

→ is included to the class which does not have a constructor when it is coming for compilation

```
class a
{
    int b;
    a() {}
    void c()
    {
        SOP(a);
    }
}
```



n/s/s/s

Destructors

Purpose:

The resources to be released when the object is no-longer in use.

↓
Java System

* garbage collection() (inbuilt method executed max.)

↑
periodically java collects

↓
for objects not in use
it will not allocate resource

We cannot call a destructor

Destructor will run at system level user will not know

class test

```
{ int a;
```

```
public int b; // same as int b;
```

```
private int c;
```

```
void setc (int i)
```

```
{ c = i;
```

```
int getc ()
```

```
{ return c;
```

```
}
```

class testnew

```
{
```

```
public static void main  
(String args[])
```

```
{
```

```
test ob = new test();
```

```
ob.a = 10;
```

```
ob.b = 20;
```

```
ob.c = 30; // via
```

```
ob.setc(30);
```

```
a = ob.getc();
```

```
SOP ("c = " + a);
```

```
}
```

ARRAY OF OBJECTS:

class employee

```
{ int a, b, c;
```

```
void Init ()
```

```
{ a = 10;
```

```
b = 20;
```

```
}
```

```
void add ()
```

```
{
```

```
c = a + b;
```

```
SOP ("c = " + c);
```

```
}
```

class my

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

```
{ employee [] emp = new employee[3];
```

↓ ↓
array name

↳ square bracket
mentioning the size

```
emp[0] = new employee();
```

```
emp[1] = new employee();
```

```
emp[2] = new employee();
```

17/7/18

METHOD OVERLOADING

Attributes of a method

- name
- parameters
- return type

method name need not be unique
we can differentiate between them
with parameters and return type

```
int add()
int add(int a)
void add(double a, double)
```

polymorphism is achievable by
method overloading

class over

```
{
void test()
{
SOP("No Param");
}
```

```
void test(int a)
```

```
{
SOP("a = " + a);
}
```

```
void test(int a, int b)
```

```
{
SOP("a + b = " + (a + b));
}
```

```
double test(double a)
```

```
{
SOP("double a = " + a);
return a;
}
```

```
}
```

class overload demo

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

```
{
over o = new over();
```

```
double res;
```

```
o.test();
```

```
o.test(10);
```

```
o.test(12);
```

O/P

No param

a = 10

a + b = 3

double a = 12.25

Java

Quiz 3

> LIST THE BITWISE LOGICAL OPERATORS:

1, 2, ^, >>, << To perform binary operation on integer

> LIST THE BOOLEAN LOGICAL OPERATORS

&&, ||, != true, false condition

> GIVE AN EXAMPLE FOR TERNARY OPERATOR

(a > b) ? c : d

> IF ELSE IF LADDED

> DIFFERENTIATE BREAK & CONTINUE

> Jump states → in loop or switch

> when the control encounters break it comes out of the loop & stops the execution there in the loop

> when control encounters continue it goes to the condition again

Object Oriented System Development OOSD

SOFTWARE ENGINEERING

> Methodology for developing computer based systems

→ Analysis ^{output} algorithm, 2Q

→ Design ^{output} flowchart

→ Coding ^{output} implementation

→ Testing ^{output} sample data samples are taken

→ Maintenance

TRADITIONAL

→ It uses the software as a collection of programs with isolated data

→ Programs focuses on the functionality of the system

OO 3Q 4Q

It views the software as the set of objects which combines the functionalities and the data, The functionalities are achieved by the programs.

An approach to system development that uses system objects to build new systems & rebuild ones