13-03-2024
Page No.
Date / /

```
class SGGS {

    public static void main() {
        S.o.p("Inside main");
        SGGS sggs1 = new SGGS();
        S.o.p("Default constructor() returns:" +sggs1);
        S.o.p("class's toString() returns:"
                + sggs1.myToString());
    }

    public String myToString() {
        return "MyToString" + getClass().getName()
        +@+Integer.toHexString(hashCode());
    }
}
```

sggs1.myToString

changes

```
S.o.p("Default cons() returns:"+sggs1);
S.o.p("class's toString() returns:"+sggs1.toString());

public String toString() {
    return "" + getClass().getName()+@
        +Integer.toHexString(hashCode());
```

also check it with

```
S.o.p("Default cons() returns:");
```

If there are two or more classes we cannot make both of them public. There will be an error that second class ex: class sggs should be declared in a file named Sggs.java.

rm *.class to remove class.

#include<stdio.h>
#include "stdio.h"

Page No.
Date / /

public class SGGS extends college {

}

here class college is in another file

error: cannot find symbol
public class SGGS extends College {
When we are changing the name of the file from college.java to C.java.

String s = "sggs"
==> sggs

String s1 = "sggs"
==> sggs

String s2 = new String("sggs");
==> sggs

s.hashcode()
==> 3528256

==> s1.hashcode()
==> -11-

s2.hashcode()
-11-
The haycode are same

s1 == s2 (here it is checking the address)
==> false

s1.equals(s2)
==> true
In equals method it is checking the hascode

14-03-2024
Page No.
Date / /

Supeer() in inheritance.

If the content of the object are same then hascode are equal. If the hascode are equal it doesn't always mean the content are equal.

```
class College {
    Steing CollegeName;

    college ( Steing collegeName){
        this.collegeName = collegeName;
    }
}

class TechUni extends college{

    Steing uniName;

    TechUni (Steing uniName){ ← Supeer();
        this.uniName = uniName;
    }
    static public void main (Steing[] args){
        TechUni Sggst = New TechUni()
        S.O.P(" ");
    }
}
```

error: constructor college in class college cannot be applied to given types.

```
TechUni (Steing UniName){
```

When subclass object is created, first of all the super class default constructor is called & then only the sub class constructor is called.

For each & every constructor by default super() this thing is there it is hidden but it is always there.
This line is implicitly there but it is hidden.
you can write it explicitly also. as follows:

```
TechUni (String uniName) {
        super();
        this.uniName = uniName;
```

super (clgName);

error: non-static variable deptName cannot be referenced from a static context.

execute it & see the result

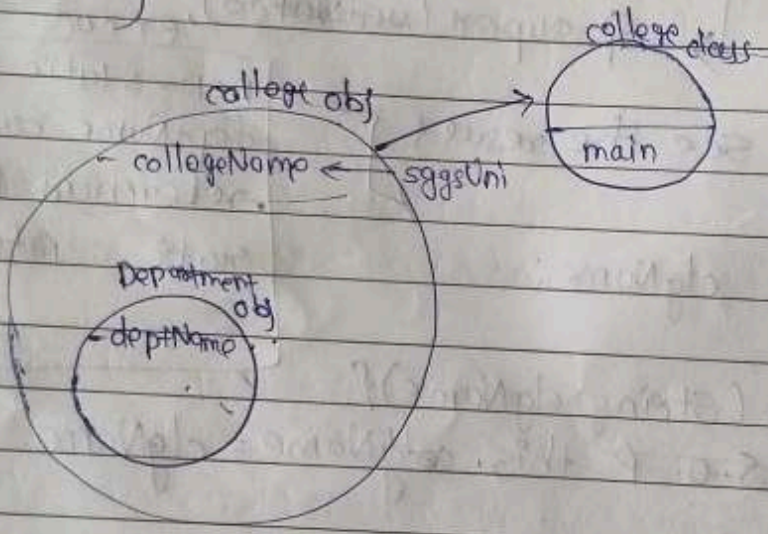
```
class College {
    String clgName;

    College (String clgName){
        this.clgName = clgName.
    }
class Department {
    String deptName;

    Department (String deptName){
        this.deptName = deptName;
    }
}
static public void main ( ) {
    College sgg = New College ("SGGSUni");
    S.O.P (collegeName)
    Department D1 = New Department ("IT Dept");
    S.O.P (    );
```

see the error here by executing.

Page No.
Date / /

```
static public void main ( ) {
    College sggst = New college ("SGGS Uni");
    S.O.P (sggst.collegeName);
    Do      dept = sggst.new Depa ("IT Dept");
    S.O.P ("" +sggst);
}
```

eee: cannot find symbol



Add this method to class Department

```
String getDeptDetails(){
    return "college Name:" + collegeName + "Depart-
                ment Name:" + deptName;
}
```

call it in main.

Page No.
Date / /

```
class college {
    Steing Name;

    college ( Steing collegeName) {
        this.name = collegeName;  or
    }                              college.this.name
class Department {                    =collegeName.
    Steing name;

        Department(Steing deptName ) {
            this.name = deptName;
        }
    Steing getDeptDetails(Steing name){
        return  name + " in \n College Name"
                + College.this.name +
                " \n Dept Name" + this.name;
                }
    S P V M ( ) {
        College sggs = New college ("SGGS Uni");
```

Page No.
Date / /

```
class SGGS {
    public static void main(String[] args) {
        System.out.println("Inside main");
        SGGS sggs1 = new SGGS();
        S.O.P("Default constructor() return:" + sggs1);
        S.O.P("class's toString() return:" +
                                    sggs1.myToString());
    }
    public String myToString() {
        return "MyToString" + getClass().getName() +
            "@" + Integer.toHexString(hashCode());
    }
}
```

output: Inside main
Default constructor() return: SGGS@1dbd16a6
class's toString() return: MyToStringSGGS@1dbd16a6

This line prints out the result of calling the 'toString()' method implicitly on the sggs1 object. Since you haven't overridden the toString() method in your class, it will use the default implementation provided by object class. this default implementation includes the class name & the object hashcode.

explicitly calls the myToString() method defined in class.

used to modify the content of mutable string

S.O.P(sggs1)

18-3-2024

Page No.

Date / /

nextInt()

for each loop

```
int arr[5] = { 1, 2, 3, 4, 5 }

for (int a : arr) {
        s.o.p(a);
    }
```

Static block.

Block of code enclosed in braces {} preceded by static keyword. When a class is loaded into memory, static block is executed only once, before the execution of main method.

Cannot have access modifiers, cannot accept any parameter

Init block (instance initialization block)

block of code within class executed when an instance of the class is creted.. not preceded by static keyword.

Executed in the order they appear in the class.

before the execution of the constructor code.

cannot have access modifiers, cannot accept parameter.

== operator is used to compare the memory address of two objects or the values of primitive typ

When used with object == checks whether two references point to the memory location whether they refer to the exact same object instance

int student[] = New int[6]

eeeee:

Page No.

Date / /

int students[] = new int[6]

==>

==> int students[6]

steing s1 = "sggs";

SteingBuffer s1 = New StingBuffer("IT");

==> sggs IT.

Steing is mutable & SteingBuffer is imutable.

Subclass provides a specific implementation of a method, that is already defined in its superclass.

```
class Animal {
    void soun d(){
        s.o.p ( Anl
    }
class dog extends Animal {

    void sound() {
        s.o.p ( Dog
}
```

Overloading → multiple method with same name but diff Paraame, diff method signature

Overridden → method have same name, return type. & Parameter as the method in superclass.

19-3-2024

Page No.

Date / /

jshell > Thread.sleep(2000);

error: unreported exception InterruptedException must be caught or declared to be thrown.

```
static void clesce(){
  try{
      Thread.sleep(1500);
    }
  catch(InterruptedException){
    }
  S.O.P("\033[H\033[2J");
public static void main(String args[]){
    clesce();
    Pragya2024._4();
```

Page No.
Date / /

Thread.sleep() method is used to pause the execution of the current thread for a specified amount of time. Useful for various purposes such as delaying execution, creating animations.
Thread.sleep() method accepts a single parameter representing the duration of time to sleep in milliseconds

Java is platform independent.
Java program are compiled into bytecode, which can run on any platform with Java virtual machine (JVM) installed, regardless of hardware or os.

Compiler ⇒ Java programming language into bytecode.

toString ⇒ return a string representation of an object [S.O.P(S1)]

StringBuffer
→ Appending strings
→ Inserting characters at specific positions
→ Replacing characters.
→ Deleting characters.

String objects are immutable, their values cannot be changed.
String is immutable you can change it using or modify it using StringBuffer and stringBuilder.

Page No.
Date / /

StringBuilder is not thread-safe., it should not be wed in multi-threaded enviornment.

super() is a keyword used to call the constructor of the superclass from the constructor of the subclass. Used to explicitly invoke the constructor of the immediate superclass.

1. Calling superclass constructor.
When you create an instance of a subclass, java automatically calls the constructor of the superclass before executing the code in the constructor of the subclass. java implicitly inserts a call to the parameterless constructor of the superclass.

2. Accessing superclass constructor with arguments.
super(arguments) ← it is allowed.

3. Placement: it must be the first statement in the subclass constructor.

Page No.
Date / /

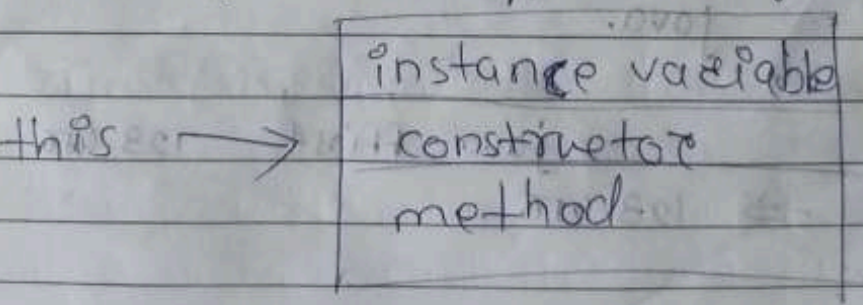
charAt ⇒ s1.charAt(5), it gives the 5th character in the string s1.

If we create an object to super class, we can access only the super class members, but not the sub class members. But if we create sub class object, all the members of both super and sub classes are available to it.

If Sometimes, the super class & sub class members may have same names, in this case by default only sub class members are available.

super.variable
super.method().

"this" is a keyword that refers to the object of the class where it is used. this refers to object of the present class. Generally we write instance variable constructor & method in a class. All these members are referred by "this". When an object is created to a class, default reference is also creted internally to the object. This default reference is nothing by "this". So "this" can refers to all the things of the present object.



void modify (int x) {
this.x = x; // store local variable x into present class instance variable x
}

2-4-2024

* Inner class.

Exception → checked
Unchecked

If we are getting exception at the time of compilation then we can check the exception.
Compile time exception.
Runtime exception → Unchecked (arrayIndexoutofbound)
→ error (StackOverflow)

Illegal operation: Divide by zero.

In the recursive call we keep on creating objects then the memory of JVM will be full & then the stackflow overflow error occurs & it is not internally in the program.

* Command-line arguments (explore this concept)
Ex: ⇒ java ffirst java ffirst
⇒ java.

java ffirst 123.456 567
⇒ 123.456

Page No.
Date / /

Parameter & Whatever you are providing in function signature

public static void main (String ... args) {

}

The above is the valid syntax.

(String args...) ← It will give an error.

Unchecked

public static void main (String ... args) {

S.O.P (args[0]);

}

It will compile but.

→ At the time of execution

error: ~~ex~~ ArrayIndexOutOfBoundException

Index 0 out of bounds for length 0.

class FFirst {

P. S. V. M (String ... args) {

try {

S.O.P (args[0]);

}

Catch (Exception e) {

S.O.P ("Exception at line Number 5");

}

}

}

Page No.
Date / /

getMessage()
getLocalizedMessage()
toString()
printStackTrace()

error comes at ~~the time~~ runtime.
or at the time of execution.

Parent class can we child class object

Exception class ~~ob~~ can make a call to method of Throwable class because Throwable is a parent class & exception class inherit the properties of throwable class And another point if it is private then we cannot access it.

error: 'try' without 'catch', 'finally' or resource declaration.

```
p.s.v.m (String ...args) {
    try {
        S.o.p(args[0]);
    }
    finally {
        S.o.p("Exception at line");
    }
}
```

checked → exception

Page No.
Date / /

error: 'catch' without 'try'

```
P.S.V.M(String ...args){
    try {
        S.O.P(args[0]);
    }
    catch {
        S.O.P("Inside catch");
    }
    finally {
        S.O.P("Inside finally");
    }
}
```

O/P ⇒ Inside catch
Inside finally

~~java Example 0~~

java classname 0x8

O/P ⇒ 0x8
Inside finally.

checked exception ⇒ error: unreported exception InterruptException; must be caught or declared to be thrown.

Date / /

```
try {
    Thread.sleep(1000)
    s.o.p (args[0]);
    Thread.sleep(1000);
  }
Catch (ArrayIndexOutofBoundsE e) {
    S.O.P ("Inside ArrayIndex — " — catch");
  }
catch (InterruptException e) {
    S.O.P("Inside Inte — " —  catch");
  }
finally {
    S.O.P("Inside finally");
  }
}
```

If i am trying to perform some activity but i fail to perform that time exception occur.

```
Catch (Exception e) {
    S.O.P("Inside Exception catch");
  }
```

Add this catch block at first & last & see the result.

```
P.S.V.M (String ..args) throws InterruptedExcep {}
```

Page No.
Date / /

Compile time error: This error occur during the compilation phase of a program.
ex: Missing semicolons, undeclared variables.

Runtime error: This error occur while a program is running.
cause: division by zero, stack overflow.
Ex: Accessing an array element beyond it's bounds.

logical error: (semantic error) when the program compiles and runs without error message, but does not produce the expected output due to flaw in the algorithm.

ex: Incorrect condition, wing coeong formula.

* Exception are runtime error.

Runtime ⇒ write main method in java without it's parameter String args[]
Runtime error are detected by JVM, only at runtime.

* The exception that are checked at compilation time by the java compiler are called checked exception.
* while the exception that are checked by JVM are called unchecked exceptions.

Unchecked exceptions and errors are considered as unrecoverable.
In case of other exception (checked) programmer should either handle them or throw them without handling them.

4-4-2024

Page No.

Date / /

method < concrete
abstract

Interfaces contain constant variable & abstract method does not contain concrete method.

Abstract class: (don't have a definition)
Abstract class may contain concrete method
concrete method means it contains class definition.

Abstrct class should contain atleast one abstract method

```
Interface Voter {
        void castVote();
}
interface EC {
        void checkValidityOfUser();
}
abstract class ECIndia implements Voter, EC {
        void castVote(){
        }
        void checkValidityOfVoter(){
        }
}
```

Page No.
Date / /

```
class Demo {
    p.s.v.M (String args[]){
        int i=0
        int j=0

        try {
            j = 18/i;
        }
        Catch (Exception e){
            S.o.P("Something went wrong" +e);
        }
        S.o.P(j);
        S.o.P("Bye");
} }
```

Catch block will be executed only if there is a exception.

```
P.s.v.M (String args[]){
    int i=0;
    int j=0;

    int nums[] = new int[5];
    String ste = null;

    try {
        j= 18/i;
        S.o.P(ste.length());
        S.o.P(nums[1]);
        S.o.P(nums[5]);
    }
```

Date / /

```
catch (ArithmeticException e)
{
    S.o.p ("cannot divide by zero");
}
Catch (ArrayIndexOutOfBoundsException e)
 {
    S.o.p ("Stay in your limit.    ");
 }
catch (Exception e){
        S.o.p ("Something went wrong" +e)
 }

S.o.p (j)
S.o.p ("Bye");

}
```

Exception is parent class so keep this at bottom.

```
class Demo {
    p.s.v.m (String args[]){
        int i=20;
        int j=0;
        try {
            j=18/i;
            if (j==0)
                throw new ArithmeticException("I
                    don't wan    ");
        }
        Catch (ArithmeticException e){
            j=18/1;
            S.o.p ("that is the default output to
        }
```

```
Catch (Exception e){
        S.o.p ("Something went wrong" +e);
    }
S.o.P(j);
S.O.P("Bye");

}
```

* Custom Exception in Java

```
class NavinException extends Exception
{
    public NavinException (String string){
        super(string);
    }
}
public class Demo

try{
    j=18/i;
    if(j==0)
    throw new NavinException("I don't want to
                print zero");
}
Catch(NavinException e){

}
```

rest of the code is same.

You can crete your own exception with Exception class. You have to pass the msg to Constructor

10-4-2024

Page No.

Date / /

~~192-11~~

error specific question.
Exception handling
interface, abstract class
file handling

Throw exception

If their is a relation betn the two things like ECIndia & ECMaharashtra then you need to go with classes.

public is a weaker access specifier.
private is the weakest

gedit
vi — terminal to exit from this command press W.

abstract class Sample

error: missing method body or declare abstract.

abstract class Sample {
    void funSampleAbstract();
}

you need use abstract keyword.
abstract void funSampleAbstract();
then it will compile.

error: interface abstract method cannot have body

work with constant variable

Page No.
Date / /

```
abstract class Sample{
        abstract void funSampleAbtract();
        void fun(){

        }
}
```

error: sample is not abstract and does not overeid abstract method funSampleAbstract() in sample.

```
class sample{
        -11-
}
```

@ you cannot create the object with abstract class.
Abstract class can be superclass.

Abstract class cannot hold reference ~~var~~ or instance.