KG REDDY
College of Engineering
& Technology

Certification Course in Computer Science and Engineering with Specialization in "OOPS THROUGH JAVA"

Held On

August 2019 to November 2019

Department of Computer Science Engineering,

KG Reddy College of Engineering & Technology

Chilkur(Village), Moinabad(Mandal), Hyderabad RR Dist-501504

Coordinator

Principal

Principal
KG Reddy College of Engineering & Technology
Chilkur (V), Moinabad (M).
R.R.Dist., Telangana.

SUMMARY REPORT ON OOPS THROUGH JAVA

About the Course

The certificate course on OOPs Through Java is concluded its work successfully by department of computer science and engineering (CSE) in KG ready college of Engineering and technology (KGRCET), Hyderabad, Telangana. This course is a forum to bring together students to discuss innovative ideas and diverse topics of this course on next generation of information technologies. Department has taken a new step for students to improve the quality of study through this course and become most wide scale , extensive, spectacular event in computer science engineering. The ten weeks course was held in two locations of the department (a) Department E-learning room for theory class and (b) Department laboratory for practical class. OOPs concepts in Java are the main ideas behind Java's Object Oriented Programming. They are an abstraction, encapsulation, inheritance, and polymorphism. Grasping them is key to understanding how Java works. Basically, Java OOP concepts let us create working methods and variables, then re-use all or part of them without compromising security.

Scope of the Course

Basically, Java is not just a Programming language but it is a programming atmosphere to develop and deploy enterprise applications. It is important for information technology industry to develop and create multiple web-based or server based applications to enhance the industrial competency. There is huge scope for this programming language. As far as the future of Java is concerned, it will be intertwined with that of agile and lean thinking that will allow the Java community to continue to innovate and deliver quality systems that address business needs. I believe that developers have a role to play in this story, by helping to accelerate the evolution of IT from isolated systems to collaborative development. It is embedded in many of the world's important IT systems and is in a good position to play a part in future innovation. The best part about Java is, its eco-system is self-sustaining, from mobility (Android) to middleware (Hadoop), it impacts everything and will continue to make a big impact in future too.

Objectives of the course

The objective of the course is to enable students to review the concepts of Java in a better and practical way and be able to write their own solutions to practical scenarios

Outcomes

This course shared the knowledge among students The main outputs are mentioned below:

CO1: Demonstrate understanding of oops concepts by writing simple java programs
CO2: Apply multithreading and exception handling concepts of java to given problems
CO3: create simple applet window.

Summary of Participants

Number of students attended this course:100
Number of students attended written exam: 100
Number of students qualified the exam: 85

Day-wise Summary
August
7/8/19 & 10/8/19

Java is one of many programming languages and technologies supported by Stackify's leading tools, Retrace and Prefix. Because at Stackify we aim to help developers become better developers, we're taking a look at some of the foundational concepts in the Java programming language. Read on for a primer on OOP concepts in Java.

Definition of OOP Concepts in Java

OOP concepts in Java are the main ideas behind Java's Object Oriented Programming. They are an abstraction, encapsulation, inheritance, and polymorphism. Grasping them is key to understanding how Java works. Basically, Java OOP concepts let us create working methods and variables, then re-use all or part of them without compromising security.

List of OOP Concepts in Java

There are four main OOP concepts in Java. These are:

Abstraction. Abstraction means using simple things to represent complexity. We all know how to turn the TV on, but we don't need to know how it works in order to enjoy it. In Java, abstraction means simple things like objects, classes, and variables represent more complex underlying code and data. This is important because it lets avoid repeating the same work multiple times.

Encapsulation. This is the practice of keeping fields within a class private, then providing access to them via public methods. It's a protective barrier that keeps the data and code safe within the class itself. This way, we can re-use objects like code components or variables without allowing open access to the data system-wide.

Inheritance. This is a special feature of Object Oriented Programming in Java. It lets programmers create new classes that share some of the attributes of existing classes. This lets us build on previous work without reinventing the wheel.

Polymorphism. This Java OOP concept lets programmers use the same word to mean different things in different contexts. One form of polymorphism in Java is method overloading. That's when different meanings are implied by the code itself. The other form is method overriding. That's when the different meanings are implied by the values of the supplied variables. See more on this below.

How OOP Concepts in Java Work

OOP, concepts in Java work by letting programmers create components that can be re-used in different ways, but still maintain security.

How Abstraction Works

Abstraction as an OOP concept in Java works by letting programmers create useful, reusable tools. For example, a programmer can create several different types of objects. These can be

variables, functions, or data structures. Programmers can also create different classes of objects. These are ways to define the objects.

For instance, a class of variable might be an address. The class might specify that each address object shall have a name, street, city, and zip code. The objects, in this case, might be employee addresses, customer addresses, or supplier addresses.

How Encapsulation Works

Encapsulation lets us re-use functionality without jeopardizing security. It's a powerful OOP concept in Java because it helps us save a lot of time. For example, we may create a piece of code that calls specific data from a database. It may be useful to reuse that code with other databases or processes. Encapsulation lets us do that while keeping our original data private. It also lets us alter our original code without breaking it for others who have adopted it in the meantime.

August
14/8/19 & 17/8/19

How Inheritance Works

Inheritance is another labor-saving Java OOP concept. It works by letting a new class adopt the properties of another. We call the inheriting class a subclass or a child class. The original class is often called the parent. We use the keyword extends to define a new class that inherits properties from an old class.

How Polymorphism Works

Polymorphism in Java works by using a reference to a parent class to affect an object in the child class. We might create a class called "horse" by extending the "animal" class. That class might also implement the "professional racing" class. The "horse" class is "polymorphic," since it inherits attributes of both the "animal" and "professional racing" class.

Two more examples of polymorphism in Java are method overriding and method overloading.

In method overriding, the child class can use the OOP polymorphism concept to override a method of its parent class. That allows a programmer to use one method in different ways depending on whether it's invoked by an object of the parent class or an object of the child class.

In method overloading, a single method may perform different functions depending on the context in which it's called. That is, a single method name might work in different ways depending on what arguments are passed to it.

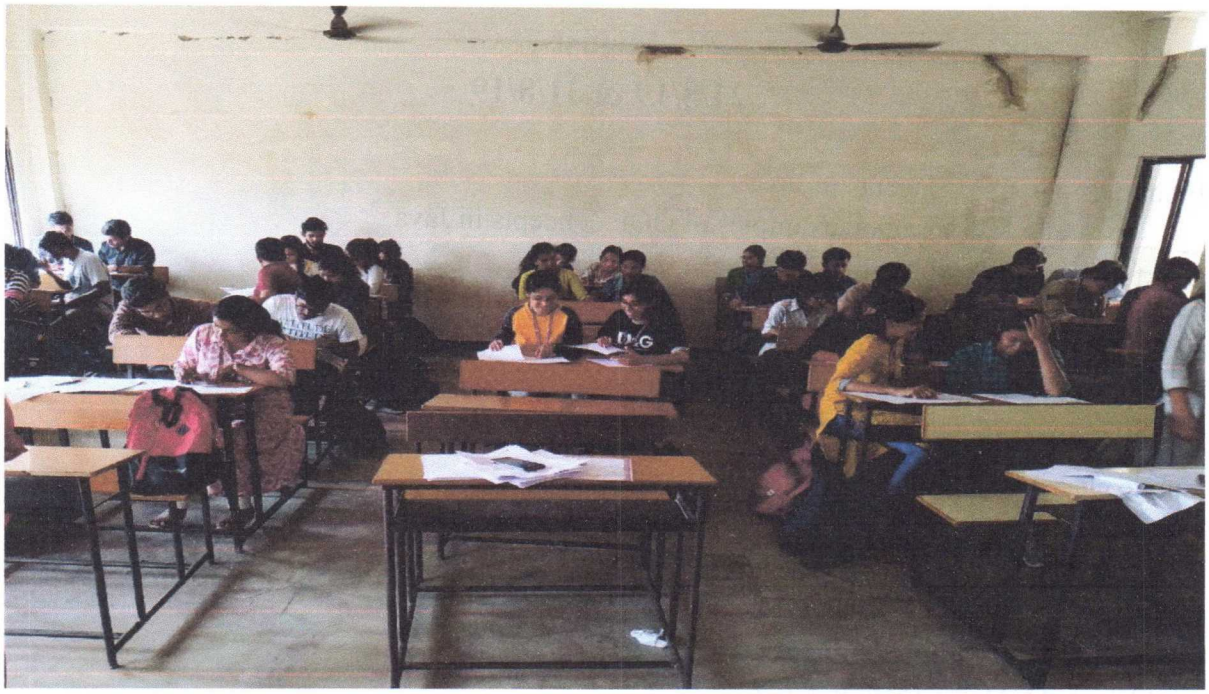
August
21/8/19 & 31/8/19

Examples of OOP Concepts in Java
Let's look at a few common examples of OOP concepts in Java.
Short Encapsulation Example in Java
In the example below, encapsulation is demonstrated as an OOP concept in Java. Here, the variable "name" is kept private or "encapsulated."

```
//save as Student.java
package com.javatpoint;
public class Student {
 private String name;
 public String getName() {
  return name;
 }
 public void setName(String name) {
  this.name = name
 }
}
//save as Test.java
package com.javatpoint;
class Test {
 public static void main(String[] args) {
  Student s = new Student();
  s.setName("vijay");
  System.out.println(s.getName());
 }
}
```

Compile By: javac -d . Test.java
Run By: java com.javatpoint.Test

Output: vijay



Students Brain storming on given problem statement

September
3/9/19 & 7/9/19

Java Access Specifiers (also known as Visibility Specifiers) regulate access to classes, fields and methods in Java.These Specifiers determine whether a field or method in a class, can be used or invoked by another method in another class or sub-class. Access Specifiers can be used to restrict access. Access Specifiers are an integral part of object-oriented programming.

Types Of Access Specifiers :

In java we have four Access Specifiers and they are listed below.

1. public
2. private
3. protected
4. default(no specifier)

public specifiers :

Public Specifiers achieves the highest level of accessibility. Classes, methods, and fields declared as public can be accessed from any class in the Java program, whether these classes are in the same package or in another package.

Example :

```
public class Demo {  // public class
public x, y, size;   // public instance variables
}
```

private specifiers :

Private Specifiers achieves the lowest level of accessibility.private methods and fields can only be accessed within the same class to which the methods and fields belong. private methods and fields are not visible within subclasses and are not inherited by subclasses. So, the private access specifier is opposite to the public access specifier. Using Private Specifier we can achieve encapsulation and hide data from the outside world.

Example :

```
public class Demo {   // public class
private double x, y;   // private (encapsulated) instance variables

public set(int x, int y) {  // setting values of private fields
this.x = x;
this.y = y;
}
public get() {  // setting values of private fields
return Point(x, y);
}
}
```

protected specifiers :
Methods and fields declared as protected can only be accessed by the subclasses in other package or any class within the package of the protected members' class. The protected access specifier cannot be applied to class and interfaces.
default(no specifier):
When you don't set access specifier for the element, it will follow the default accessibility level. There is no default specifier keyword. Classes, variables, and methods can be default accessed.Using default specifier we can access class, method, or field which belongs to same package,but not from outside this package.

Example :

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

September
17/9/19 & 21/9/19

Example of Inheritance in Java
It's quite simple to achieve inheritance as an OOP concept in Java. Inheritance can be as easy as using the extends keyword:

```
class Mammal {

}
class Aardvark extends Mammal {

}
```

Short Example of Polymorphism in Java
In the example below of polymorphism as an OOP concept in Java, we have two classes: Person and Employee. The Employee class inherits from the Person class by using the keyword extends. Here, the child class overrides the parent class. For the full example, see this blog post.

```
class Person {
 void walk() {
  System.out.println("Can Run....");
 }
}
class Employee extends Person {
 void walk() {
  System.out.println("Running Fast...");
 }
 public static void main(String arg[]) {
  Person p = new Employee(); //upcasting
  p.walk();
 }
}
```

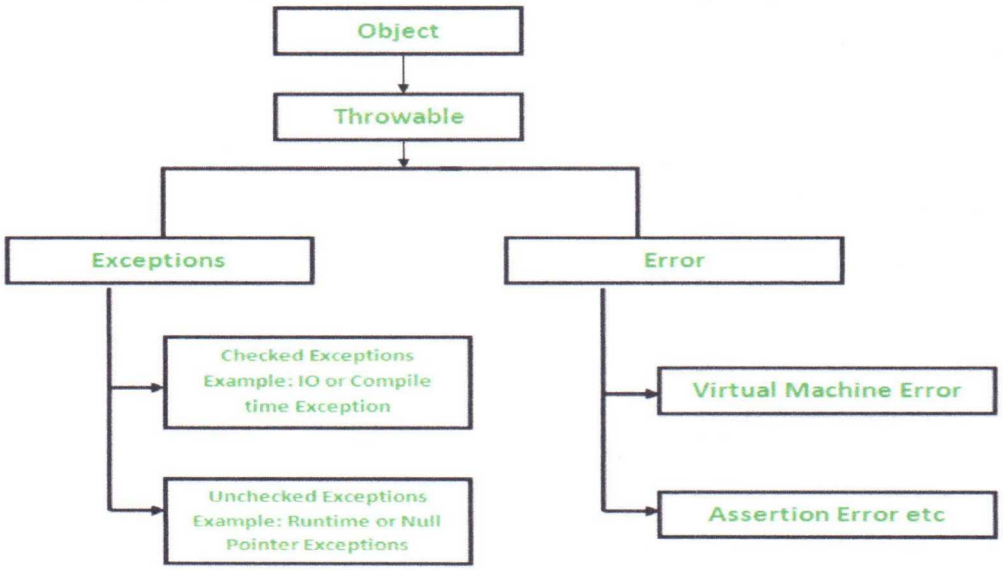
September
25/9/19 &28/9/19

What is an Exception?
An exception is an unwanted or unexpected event, which occurs during the execution of a program i.e at run time, that disrupts the normal flow of the program's instructions.
Error vs Exception

Error: An Error indicates serious problem that a reasonable application should not try to catch.
Exception: Exception indicates conditions that a reasonable application might try to catch.

Exception Hierarchy
All exception and errors types are sub classes of class Throwable, which is base class of hierarchy.One branch is headed by Exception. This class is used for exceptional conditions that user programs should catch. NullPointerException is an example of such an exception.Another branch,Error are used by the Java run-time system(JVM) to indicate errors having to do with the run-time environment itself(JRE). StackOverflowError is an example of such an error. stu

Students writing important concepts



How JVM handle an Exception?
Default Exception Handling: Whenever inside a method, if an exception has occurred, the method creates an Object known as Exception Object and hands it off to the run-time system(JVM). The exception object contains name and description of the exception, and current state of the program where exception has occurred. Creating the Exception Object and handling it to the run-time system is called throwing an Exception. There might be the list of the methods that had been called to get to the method where exception was occurred. This ordered list of the methods is called Call Stack. Now the following procedure will happen.

The run-time system searches the call stack to find the method that contains block of code that can handle the occurred exception. The block of the code is called Exception handler.
The run-time system starts searching from the method in which exception occurred, proceeds through call stack in the reverse order in which methods were called.

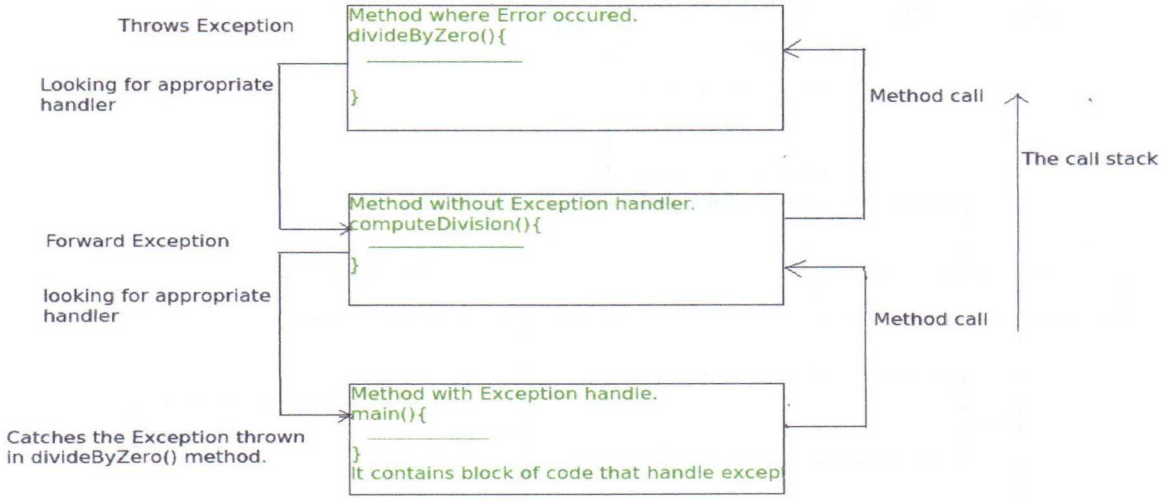
If it finds appropriate handler then it passes the occurred exception to it. Appropriate handler means the type of the exception object thrown matches the type of the exception object it can handle.

If run-time system searches all the methods on call stack and couldn't have found the appropriate handler then run-time system handover the Exception Object to default exception handler, which is part of run-time system. This handler prints the exception information in the following format and terminates program abnormally.

Exception in thread "xxx" Name of Exception : Description

... . // Call Stack

See the below diagram to understand the flow of the call stack.



The call stack and searching the call stack for exception handler.

October
23/10/19 & 26/10/19:

Example :

filter_none

edit

play_arrow

brightness_4

```
// Java program to demonstrate how exception is thrown.
class ThrowsExecp{

    public static void main(String args[]){

        String str = null;
        System.out.println(str.length());
```

```
    }
}
```

Output :

```
Exception in thread "main" java.lang.NullPointerException
    at ThrowsExecp.main(File.java:8)
```

Let us see an example that illustrate how run-time system searches appropriate exception handling code on the call stack :

```
filter_none
edit
play_arrow
brightness_4
// Java program to demonstrate exception is thrown
// how the runTime system searches th call stack
// to find appropriate exception handler.
class ExceptionThrown
{
    // It throws the Exception(ArithmeticException).
    // Appropriate Exception handler is not found within this method.
    static int divideByZero(int a, int b){

        // this statement will cause ArithmeticException(/ by zero)
        int i = a/b;

        return i;
    }

    // The runTime System searches the appropriate Exception handler
    // in this method also but couldn't have found. So looking forward
    // on the call stack.
    static int computeDivision(int a, int b) {

        int res =0;

        try
        {
          res = divideByZero(a,b);
        }
        // doesn't matches with ArithmeticException
        catch(NumberFormatException ex)
        {
          System.out.println("NumberFormatException is occured");
        }
```

```
        return res;
    }

    // In this method found appropriate Exception handler.
    // i.e. matching catch block.
    public static void main(String args[]){

        int a = 1;
        int b = 0;

        try
        {
            int i = computeDivision(a,b);

        }

        // matching ArithmeticException
        catch(ArithmeticException ex)
        {
            // getMessage will print description of exception(here / by zero)
            System.out.println(ex.getMessage());
        }
    }
}
```

October

30/10/19 & 2/11/19

Multithreading in Java

Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU. Each part of such program is called a thread. So, threads are light-weight processes within a process.

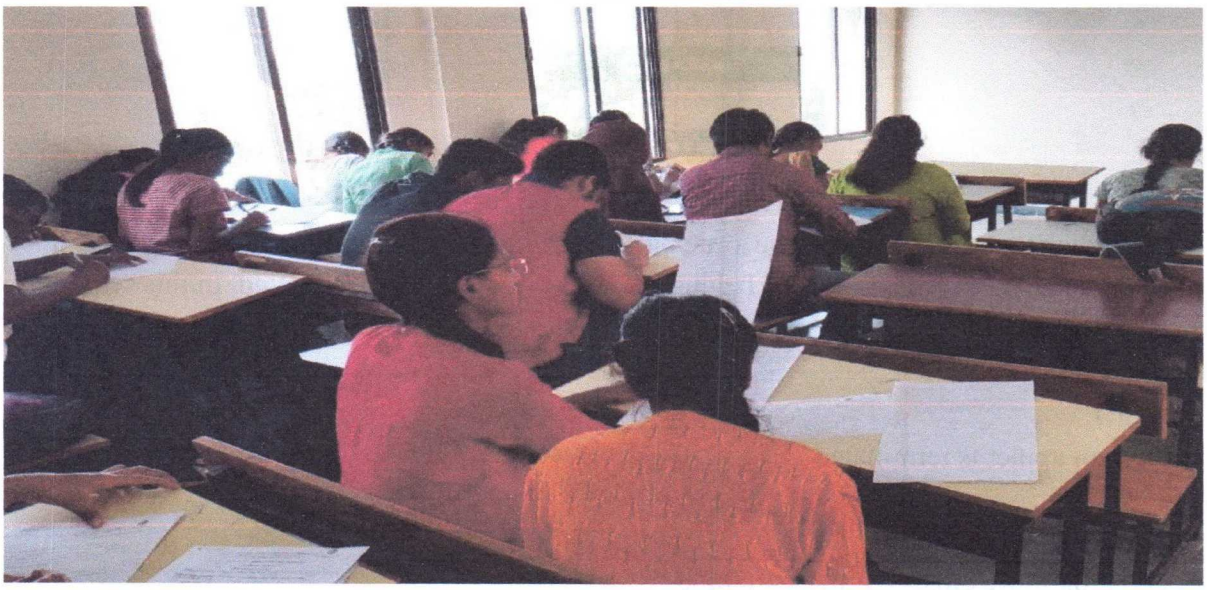
Threads can be created by using two mechanisms :

1. Extending the Thread class
2. Implementing the Runnable Interface

Students in E-Class room

Thread creation by extending the Thread class

We create a class that extends the java.lang.Thread class. This class overrides the run() method available in the Thread class. A thread begins its life inside run() method. We create an object of our new class and call start() method to start the execution of a thread. Start() invokes the run() method on the Thread object.

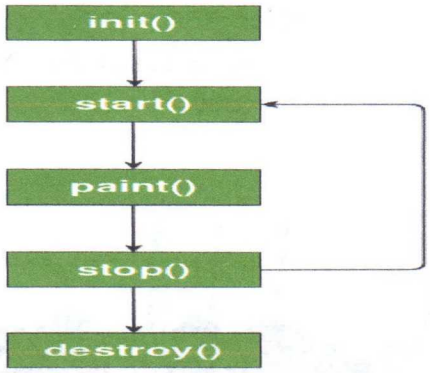


Students writing solution to the given problem statement

November
13/11/19 & 16/11/19

What is Applet?
An applet is a Java program that can be embedded into a web page. It runs inside the web browser and works at client side. An applet is embedded in an HTML page using the APPLET or OBJECT tag and hosted on a web server.
Applets are used to make the web site more dynamic and entertaining.
Important points :
All applets are sub-classes (either directly or indirectly) of java.applet.Applet class.
Applets are not stand-alone programs. Instead, they run within either a web browser or an applet viewer. JDK provides a standard applet viewer tool called applet viewer.
In general, execution of an applet does not begin at main() method.
Output of an applet window is not performed by System.out.println(). Rather it is handled with various AWT methods, such as drawString().

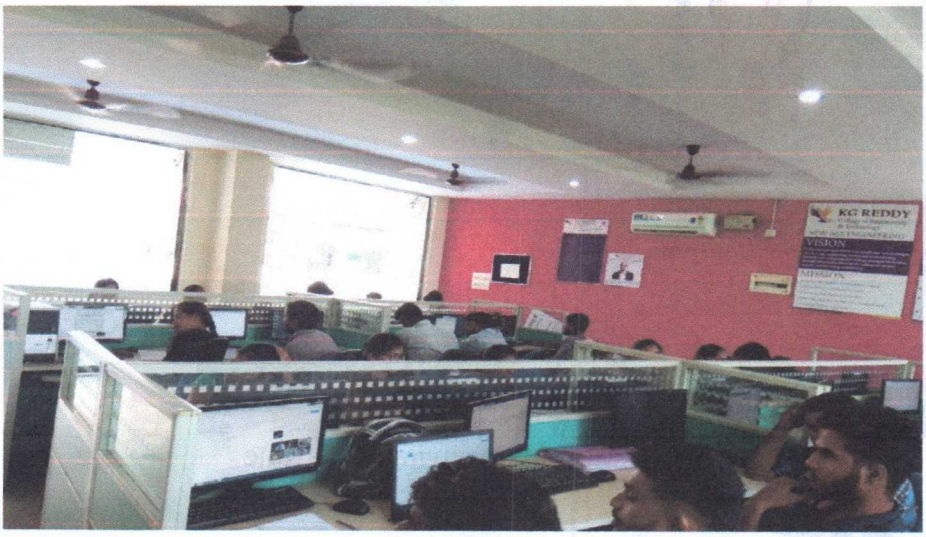
Life cycle of an applet :



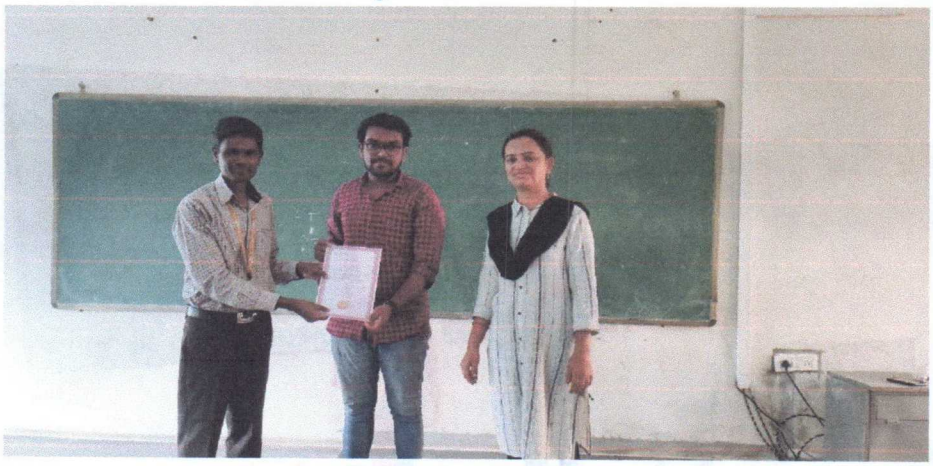
It is important to understand the order in which the various methods shown in the above image are called. When an applet begins, the following methods are called, in this sequence:
init( )
start( )
paint( )
When an applet is terminated, the following sequence of method calls takes place:
stop( )
destroy( )
Let's look more closely at these methods.
init( ) : The init( ) method is the first method to be called. This is where you should initialize variables. This method is called only once during the run time of your applet.
start( ) : The start( ) method is called after init( ). It is also called to restart an applet after it has been stopped. Note that init( ) is called once i.e. when the first time an applet is loaded

whereas start( ) is called each time an applet's HTML document is displayed onscreen. So, if a user leaves a web page and comes back, the applet resumes execution at start( ).

paint( ) : The paint( ) method is called each time an AWT-based applet's output must be redrawn. This situation can occur for several reasons. For example, the window in which the applet is running may be overwritten by another window and then uncovered. Or the applet window may be minimized and then restored.

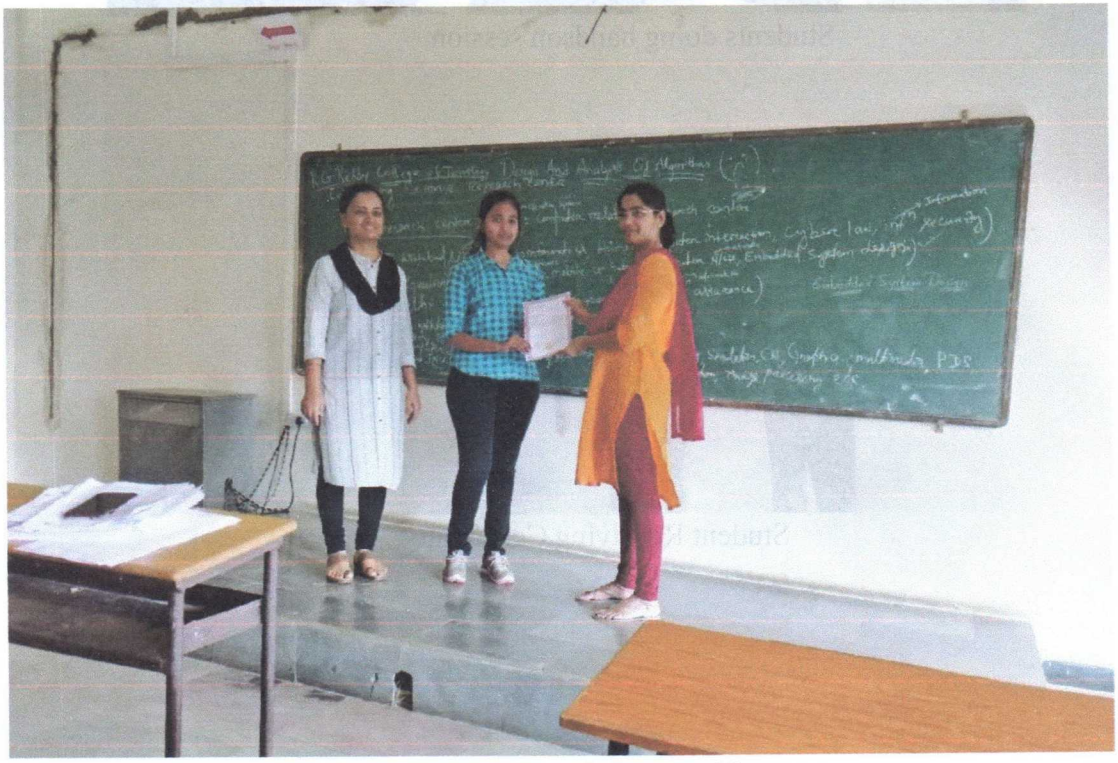


Students doing handson session



Student Receiving Certificate

Student Receiving Certificate



Student Receiving Certificate

KG REDDY
College of Engineering
& Technology

Accredited by NAAC

Ref No: KGRCET/CSE/2019-20/

Date: 17/07/2019

CIRCULAR

All the students of III-Year I-semester B.Tech CSE are here by informed to enroll for the certification course on "OOPs through Java", which is offered by KG Reddy college of Engineering and Technology from August 2019 to November 2019. Interested students are instructed to contact Dr. Hemanta Kumar Bhuyan for completing their registration before 01/08/2019.

HOD

HEAD
DEPT. OF COMPUTER SCIENCE & ENGINEERING
K.G. REDDY COLLEGE OF ENGINEERING & TECHNOLOGY
CHILKUR (V), MOINABAD, R.R. DIST.501 504.

Principal

Principal
KG Reddy College of Engineering & Technology
Chilkur (V), Moinabad (M).
R.R.Dist., Telangana

Syllabus

Week 1 :	Overview of Object-Oriented Programming and Java	pre requisites for programming
		difference between c ,C++,Java
		importance of object oriented programming
		brushing up of oops concepts
Week 2 :	Java Programming Elements	basic program in java
		using operaters in java
		class concepts
		creation of objects in java
Week 3 :	Input-Output Handling in Java	different input methods in java
		different output methods in java
		example programms
Week 4 :	Encapsulation	need for encapsulation
		public
		private
		protected
		programs using different acces specifiers
Week 5 :	Inheritance	reusability withinheritance
		different types of inheritance
		sub class super class and accesibity
Week 6 :	Exception Handling	complile time exceptions
		run time exceptions
		try catch
		throw
		finally
Week 7 :	Multithreaded Programming	importance of multi threading
		thread states
		methods of thread class
		examples

Week 8	: Java Applets and Servlets	introduction and differences
		life cycle
		packages
		examples
Week 9 :	Java Swing and Abstract Windowing Toolkit (AWT)	introductiuon to swings
		AWT
		difference
		usage of different API
		examples of swings & AWT
Week 10 :	Networking with Java	java.net package
		examples of networking applications
Week 11:	Java Object Database Connectivity (ODBC)	jdbc vs Odbc
		API's
		steps for database connectivity
		examples
Week 12:	Interface and Packages for Software Development	introduction
		importing of packages and interfaces
		diffreneces
		examples

Syllabus Coverage with dates

Week 1	Introduction and differences	7/8/19 & 10/8/19
Week 2	Operators, classes and objects of java.	14/8/19 & 17/8/19
Week 3	Input output methods of java and programs, hands on in lab	21/8/19 & 31/8/19
Week 4	Access specifiers and their use in java, examples programs	3/9/19 & 7/9/19
Week 5	Inheritance ,types, reusability examples program, polymorphism example	17/9/19 & 21/9/19
Week 6	Exception handling in java, different key words of java their functioning examples	25/9/19 &28/9/19
Week 7	Exception handling examples explanation	23/10/19 & 26/10/19
Week 8	Multithreading in java thread class extension.	30/10/19 & 2/11/19
Week 9	Different random examples	6/11/19 & 9/11/19
Week 10	Java applet life cycle applet examples	13/11/19 & 16/11/19

Attendance

SECTION -A

S.No	ROLL NO	NAME	10-Aug	17-Aug	31-Aug	07-Sep	21-Sep	28-Sep	26-Oct	02-nov	09-Nov	16-Nov
1	17QM1A0501	A Bhanu Priya	p		p	p		p	p	p	p	p
2	17QM1A0502	Addisherla Pooja Shreni			p	p				p	p	
3	17QM1A0503	Adusumilli Naga Sai Poojitha		p	p	p			p	p	p	
4	17QM1A0504	Ajit Kulkarni		p				p	p			
5	17QM1A0505	Akkaladevi Sandeep		p					p			
6	17QM1A0506	Amrabad Shiva Bharghav				p		p			p	
7	17QM1A0507	Arelly Sri Chaithanya Kumar				p			p		p	
8	17QM1A0508	Avula Neha		p	p	p		p		p	p	
9	17QM1A0509	Baddam Srujan Reddy	p			p		p	p		p	p
10	17QM1A0510	Balreddy Madhuri		p		p			p		p	
11	17QM1A0511	Banngar Nabi	p	p	p	p			p	p	p	p
12	17QM1A0512	Bathula Pavan Kumar Goud							p			
13	17QM1A0513	Bellamgari Pranavi		p	p	p			p	p	p	
14	17QM1A0514	Biradar Jaish		p		p		p	p		p	
15	17QM1A0515	Bokka Sai Kiran Reddy						p	p			
16	17QM1A0516	Chanda Varun Raj		p		p			p		p	

17	17QM1A0517	Chelimela Sanjay	p						p			p
18	17QM1A0518	Chintal Vijay Kumar			p	p		p	p	p	p	
19	17QM1A0519	Chirag Keerthi Sree		p		p	p		p		p	
20	17QM1A0520	Chirumamilla Raja Karthik				p		p	p		p	
21	17QM1A0521	Chityala Maheshwari	p	p	p	p		p	p	p	p	p
22	17QM1A0522	D Keerthi		p		p		p	p		p	
23	17QM1A0523	D Vivek		p	p				p	p		
24	17QM1A0525	Danturi Harshitha		p				p	p			
25	17QM1A0526	Duppekar Vyshnavi	p	p	p	p			p	p	p	p
26	17QM1A0527	Dyavarashetty Goutham		p		p			p		p	
27	17QM1A0528	G S Vaishnavi		p	p	p		p	p	p	p	
28	17QM1A0529	G Sai Pradhiksha		p		p	p		p		p	
29	17QM1A0530	Gaddam Shirisha	p	p	p	p		p	p	p	p	p
30	17QM1A0532	Sudugu Supriya		p	p	p			p	p	p	
31	17QM1A0533	Gangala Yamini	p	p		p		p	p		p	p
32	17QM1A0534	Gangavarapu Shanya Psalms	p	p	p	p		p	p	p	p	p
33	17QM1A0535	Gedela Srujana			p	p		p	p	p	p	
34	17QM1A0536	Gouravarapu Jayavenkatasai Gopi Aravind		p	p	p				p	p	
35	17QM1A0537	Gouse M D			p	p		p	p	p	p	
36	17QM1A0538	Gudguntla Naveen Kumar				p		p	p		p	

37	17QM1A0539	Ghanapuram Shekar Reddy		p		p		p	p		p	
38	17QM1A0540	Gundaveni Sai Charan			p	p			p	p	p	
39	17QM1A0541	Gundu Jalaja Reddy			p	p	p	p	p	p	p	
40	17QM1A0542	Hrithik Singh	p	p		p		p	p		p	p
41	17QM1A0543	Jilla Manideep	p		p					p		p
42	17QM1A0544	Kadari Roopesh	p	p	p	p				p	p	p
43	17QM1A0545	Kadari Vasavi		p	p	p			p	p	p	
44	17QM1A0546	Kalpaguri Vinay Kumar Gupta	p	p	p	p		p	p	p	p	p
45	17QM1A0547	Kalwa Sai Charan		p	p	p		p	p	p	p	
46	17QM1A0549	Karsai Srisailam Yadav		p		p			p		p	
47	17QM1A0550	Katipelly Praneeth Reddy		p	p	p			p	p	p	
48	17QM1A0551	Kavya Dyasapally	p		p	p		p		p	p	p
49	17QM1A0553	Kotha Abhinav Reddy	p		p	p			p	p	p	p
50	17QM1A0554	Lalchand Gurjar		p	p	p			p	p	p	
51	17QM1A0555	Laxmi Venkata Lahari N		p	p	p			p	p	p	

SECTION -B

S.No	ROLL NO	NAME	07-Aug	14-Aug	21-Aug	03-Sep	17-Sep	25-Sep	23-Oct	30-Oct	06-Nov	13-Nov
2	167B1A0508	Plsati Prashanth Reddy	P	p			p	p	p	p	p	
3	16QM1A0519	Chiluguraju Umasheshank	P	p		p	p	p	p	p	p	p
4	16QM1A0520	Chinnapesari Srikanth Reddy	P	p	p	p	p		p	p	p	p
5	17QM1A0556	M Manoj Kumar	P	p	p	p	p	p		p	p	p
6	17QM1A0557	Mamidi Madhu	P	p		p		p		p	p	p
7	17QM1A0558	Mangalagiri Krishna Sai	P			p	p	p	p			p
8	17QM1A0559	Maravapally Maharshi Reddy	P			p	p	p	p	p	p	p
9	17QM1A0560	Maregouni Priyanka			p		p		p			
10	17QM1A0562	Mettu Anusha Reddy	P	p	p	p	p	p	p	p	p	p
11	17QM1A0563	Mir Furqaan Ali	P			p	p	p	p	p	p	
12	17QM1A0564	Mogili Sriram Krupal	P	p		p	p	p		p	p	p
13	17QM1A0565	Mohammed Saifuddin	P	p		p	p		p	p	p	p
14	17QM1A0566	Mohammed Shahed	P	p		p	p	p	p	p		p
15	17QM1A0567	Mohd Jameel Ahmed	P	p		p	p	p	p	p	p	p
16	17QM1A0568	Mourya Kunal Ashokkumar	P			p	p		p	p	p	p
17	17QM1A0569	P Abhilash Reddy	P			p	p	p	p	p		
18	17QM1A0570	Permishetty Yeshwanth	P			p	p	p		p	p	p
19	17QM1A0572	Puna Prachi	P	p	p	p	p	p				p
20	17QM1A0573	R Prakash Reddy	P	p	p	p		p	p	p	p	p
21	17QM1A0574	Rameshwaram Satish Kumar	P	p		p	p		p	p	p	p
22	17QM1A0575	Renukuntla Rahul	P	p		p		p	p	p	p	p
23	17QM1A0576	S Soudamini	P			p		p	p	p	p	p
24	17QM1A0577	Sama Vani	P	p		p	p	p	p	p		p

25	17QM1A0579	Sappati Vasavi	P	p	p	p		p		p	p	p
26	17QM1A0580	Sarikonda Naveen Reddy	P	p		p		p	p	p		p
27	17QM1A0581	Shaik Altaf Hussain	P	p		p	p	p		p	p	p
28	17QM1A0582	Shankuri Harika	P			p	p	p	p	p	p	
29	17QM1A0583	Shubam Kumar	P	p		p	p	p		p	p	p
30	17QM1A0585	Sulge Nikitha	P	p		p	p	p	p	p	p	p
31	17QM1A0586	Talatoti Charles	P	p	p	p	p	p	p	p	p	p
32	17QM1A0587	Telugu Sai Vinay	P	p		p	p		p	p	p	p
33	17QM1A0588	Troopbazar Akhila	P	p		p	p	p	p	p	p	
34	17QM1A0589	V Pavani	P			p		p		p	p	p
35	17QM1A0590	V V Sai Tejaswi	P	p	p	p	p	p	p	p		p
36	17QM1A0591	Vallabaneni Sai Nikitha	P		p	p	p	p		p	p	p
37	17QM1A0593	Vishal Gupta	P	p		p	p		p	p	p	p
38	17QM1A0594	Vundyala Sreenidhi Reddy		p		p	p	p		p	p	
39	17QM1A0595	Yeleti Pranitha	P	p			p	p	p		p	p
40	17QM1A0596	Yempathi Aashritha		p			p	p	p			
41	17QM1A0597	Yogesh Singh	P	p		p	p			p		
42	17QM1A0598	Nethula Rahul Yadav	P	p		p	p	p	p	p	p	p
43	17QM1A05A0	Sd Mustaq	P	p			p	p	p		p	p
44	17QM1A05A2	Tipparamoni Bhuvaneshwar				p	p		p			
45	17QM1A05A3	Shetti Anusha	P	p		p	p	p	p		p	p
46	17QM1A05A4	Alli Sirisha		p		p	p	p	p	p		p
47	17QM1A05A5	Peechari Karunakar Reddy	P	p		p	p	P	p	p	p	p
48	17QM1A05A6	Meka Lavyasri	P	p				P	p	p	p	
49	17QM1A05B0	D Amardeep Reddy	P	p		p	p		p	p	p	p
50	18QM5A0501	Kasoju Bavana	P	p	p	p		P	p	p	p	p

KG REDDY College of Engineering & Technology

KG Reddy College of Engineering & Technology

(Approved by AICTE, New Delhi, Affiliated to JNTUH, Hyderabad) **Accredited by NAAC**
Chilkur (Village), Moinabad (Mandal), R. R Dist, TS-501504

Department of Computer Scienece and Engineering

Certificate course on

OOPS THROUGH JAVA

26/30

NAME A. Bhanu Priya **HALL TICKET NO**

1	7	8	M	1	A	0	5	0	1

Answer all the questions. All questions carry equal marks. Time: 30min. 30 marks.
FILL IN THE BLANKS:

1. The Key word used for inheritance is Extends
2. Java was initially was developed in the year 1995.
3. Constructor should have Access modifier
4. AWT stands for Abstract Window Toolkit
5. Class is a collection of object.
6. The syntax for Package is Package packagename1. packagename2;
7. Static Binding is done at compile time.
8. In Java arrays are objects.
9. The Escape character which is not available in java is backslash(\)
10. What will be return type of a method that not returns any value void.
11. What is Math. Floor (3.6)? 3.0.
12. The default value of a static integer variable of a class in java is 0.
13. Multiple inheritance means Where a class inherit more than one parent class
14. To prevent any method from overriding we declare the method as static.
15. The feel in an interface are implicitly specified as interfaces
16. All exception types are sub classes of the built in class of throwable
17. When an overridden method is called from within a subclass, it will always refer to the version that method defined by the subclass.
18. In java objects are passed as instances
19. An Applet can be viewed using AWT.

20. Members of the class specifies as private are accessible only to methods of that class.
21. Java compiler java c translates java source in to byte code file.
22. source code are used to document a program and improved its readability.
23. In Java, a character constants value is its integer value in the ASCII character set.
24. In Java a try block should immediately be followed by one or more catch blocks.
25. An Abstract data type typically comprises a value and a set of operations respectively.
26. The sequence of instruction is called a algorithm
27. In java thread can be created by implemented runnable interface & overriding the run() start()
28. When a class extends the thread class, it should override run() method of thread class to start that thread.
29. Java is a object oriented programming language.
30. System class is define in standard. I/O

KG REDDY College of Engineering & Technology

KG Reddy College of Engineering & Technology

(Approved by AICTE, New Delhi, Affiliated to JNTUH, Hyderabad)
Chilkur (Village), Moinabad (Mandal), R. R Dist, TS-501504

Accredited by NAAC

24

Department of Computer Scienece and Engineering

Certificate course on

OOPS THROUGH JAVA

NAME K-Bavana chary **HALL TICKET NO**

1	8	Q	m	5	A	0	5	0	1

Answer all the questions. All questions carry equal marks. Time: 30min. 30 marks.

FILL IN THE BLANKS:

1. The Key word used for inheritance is to extent.
2. Java was initially was developed in the year 1995.
3. Constructor should have name name as the class.
4. AWT stands for Abstract window Toolkit
5. Class is a collection of objects.
6. The syntax for Package is package name of package;
7. Static Binding is done at compile time.
8. In Java arrays are objects.
9. The Escape character which is not available in java is output.
10. What will be return type of a method that not returns any value void.
11. What is Math. Floor (3.6)? 3.0
12. The default value of a static integer variable of a class in java is 0, 0+1
13. Multiple inheritance means inherits characteristics of parent class
14. To prevent any method from overriding we declare the method as subclass.
15. The feel in an interface are implicitly specified as static, default and final
16. All exception types are sub classes of the built in class of throwable.
17. When an overridden method is called from within a subclass, it will always refer to the version that method defined by the subclass.
18. In java objects are passed as value.
19. An Applet can be viewed using HTML.

20. Members of the class specifies as private are accessible only to methods of that class.
21. Java compiler java c translates java source in to languages.
22. Comments are used to document a program and improved its readability.
23. In Java, a character constants value is its integer value in the Catch character set.
24. In Java a try block should immediately be followed by one or more _____ blocks.
25. An Abstract data type typically comprises a mathematical and a set of data respectively.
26. The sequence of instruction is called a Algorithm.
27. In java thread can be created by Extending thread class
28. When a class extends the thread class, it should override run() method of thread class to start that thread.
29. Java is a structured programming language
30. System class is define in Java.lang.package