```
// 1.Java program to Compare two strings strcmp

public class GFG {
      public static void main(String args[])
      {
            String string1 = new String("Data");
            String string2 = new String("Data");
            String string3 = new String("World");
            String string4 = new String("Java");

            // Comparing for String 1 != String 2
            System.out.println("Comparing " + string1 + " and " + string2
+ " : " + string1.equals(string2));

            // Comparing for String 3 = String 4
            System.out.println("Comparing " + string3 + " and " + string4
+ " : " + string3.equals(string4));
      }
}

// 2.strcat

public class Test {

   public static void main(String args[]) {
      String s = "Strings are immutable";
      s = s.concat(" all the time");
      System.out.println(s);
   }
}

// 3.strcpy

public class strcpy
{
      public static void main(String args[])
      {
            String s1, s2;
            s1 = new String("hello");
            s2 = s1; // This only copies s1 to s2. Am I right?

            s1="adsfsdaf";

            System.out.println(s2);
            System.out.println(s1);
      }
}

// 4.strlen

public class LengthExample
{
      public static void main(String args[])
      {
            String s1="HelloWorld";
            String s2="HiJava";
            System.out.println("string length is: "+s1.length());//the
length of javatpoint string
```

```
            System.out.println("string length is: "+s2.length());//the
length of python string
      }
}

// 5.strrev

import java.io.*;
import java.util.Scanner;

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

            String str= "Hello", nstr="";
            char ch;

      System.out.print("Original word: ");
      System.out.println("Hello"); //Example word

      for (int i=0; i<str.length(); i++)
      {
            ch= str.charAt(i); //extracts each character
            nstr= ch+nstr; //adds each character in front of the existing
string
      }
      System.out.println("Reversed word: "+ nstr);
      }
}

// 6.simple class

public class Main {
  int x = 5;

  public static void main(String[] args) {
    Main myObj = new Main();
    System.out.println(myObj.x);
  }
}

// 7.member variable and member function

import java.io.*;
public class Employee {

   public String name;

   private double salary;

   public Employee (String empName) {
      name = empName;
   }

   public void setSalary(double empSal) {
      salary = empSal;
   }

   public void printEmp() {
      System.out.println("name  : " + name );
      System.out.println("salary :" + salary);
   }

   public static void main(String args[]) {
      Employee empOne = new Employee("Rajat");
      empOne.setSalary(82000);
      empOne.printEmp();
   }
}

// 8.enum in java

public class Main {
  enum Level {
    LOW,
    MEDIUM,
    HIGH
  }

  public static void main(String[] args) {
    Level myVar = Level.MEDIUM;
    System.out.println(myVar);
  }
}

// 9.single inheritance

class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class TestInheritance{
public static void main(String args[]){
Dog d=new Dog();
d.bark();
d.eat();
}}

// 10.multilevel inheritance

class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class BabyDog extends Dog{
void weep(){System.out.println("weeping...");}
}
class TestInheritance2{
public static void main(String args[]){
BabyDog d=new BabyDog();
d.weep();
d.bark();
d.eat();
```

```
}}

// 11.hierarchical inheritance
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class Cat extends Animal{
void meow(){System.out.println("meowing...");}
}
class TestInheritance3{
public static void main(String args[]){
Cat c=new Cat();
c.meow();
c.eat();
//c.bark();//C.T.Error
}}

// 12.multiple not possible

// 13.Java Program to create and call a default constructor
class Bike1{
//creating a default constructor
Bike1(){System.out.println("Bike is created");}
//main method
public static void main(String args[]){
//calling a default constructor
Bike1 b=new Bike1();
}
}

// 14.Let us see another example of default constructor

//which displays the default values
class Student3{
int id;
String name;
//method to display the value of id and name
void display(){System.out.println(id+" "+name);}

public static void main(String args[]){
//creating objects
Student3 s1=new Student3();
Student3 s2=new Student3();
//displaying values of the object
s1.display();
s2.display();
}
}

// 15.Java Program to demonstrate the use of the parameterized
constructor.

class Student4{
    int id;
    String name;
    //creating a parameterized constructor
    Student4(int i,String n){
    id = i;
    name = n;
    }
    //method to display the values
    void display(){System.out.println(id+" "+name);}

    public static void main(String args[]){
    //creating objects and passing values
    Student4 s1 = new Student4(111,"Karan");
    Student4 s2 = new Student4(222,"Aryan");
    //calling method to display the values of object
    s1.display();
    s2.display();
   }
}

// 16.java destructor

public class DestructorExample
{
public static void main(String[] args)
{
DestructorExample de = new DestructorExample ();
de.finalize();
de = null;
System.gc();
System.out.println("Inside the main() method");
}
protected void finalize()
{
System.out.println("Object is destroyed by the Garbage Collector");
}
}

// 17.run time polymorphism in java

class Bike{
  void run(){System.out.println("running");}
}
class Splendor extends Bike{
  void run(){System.out.println("running safely with 60km");}

  public static void main(String args[]){
    Bike b = new Splendor();//upcasting
    b.run();
  }
}

// 18.operator overloading

class OverloadingExample{
static int add(int a,int b){return a+b;}
static int add(int a,int b,int c){return a+b+c;}
}

// 19.function overriding
```

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void eat(){System.out.println("eating bread...");}
}

// 20.friend function in java

public class A {
    private int privateInt = 31415;

    public class SomePrivateMethods {
        public int getSomethingPrivate() { return privateInt;  }
        private SomePrivateMethods() { } // no public constructor
    }

    public void giveKeyTo(B other) {
        other.receiveKey(new SomePrivateMethods());
    }
}

public class B {
    private A.SomePrivateMethods key;

    public void receiveKey(A.SomePrivateMethods key) {
        this.key = key;
    }

    public void usageExample() {
        A anA = new A();

        // int foo = anA.privateInt; // doesn't work, not accessible

        anA.giveKeyTo(this);
        int fii = key.getSomethingPrivate();
        System.out.println(fii);
    }
}

// 21.virtual function

class Parent {
void v1() //Declaring function
{
System.out.println("Inside the Parent Class");
}
}
public class Child extends Parent{
            void v1() // Overriding function from the Parent class
            {
            System.out.println("Inside the Child Class");
            }
            public static void main(String args[]){
            Parent ob1 = new Child(); //Refering the child class object
using the parent class
            ob1.v1();
```

```
          }
          }

// 22. stack in java

// Java code for stack implementation

import java.io.*;
import java.util.*;

class Test
{
	// Pushing element on the top of the stack
	static void stack_push(Stack<Integer> stack)
	{
		for(int i = 0; i < 5; i++)
		{
			stack.push(i);
		}
	}

	// Popping element from the top of the stack
	static void stack_pop(Stack<Integer> stack)
	{
		System.out.println("Pop Operation:");

		for(int i = 0; i < 5; i++)
		{
			Integer y = (Integer) stack.pop();
			System.out.println(y);
		}
	}

	// Displaying element on the top of the stack
	static void stack_peek(Stack<Integer> stack)
	{
		Integer element = (Integer) stack.peek();
		System.out.println("Element on stack top: " + element);
	}

	// Searching element in the stack
	static void stack_search(Stack<Integer> stack, int element)
	{
		Integer pos = (Integer) stack.search(element);

		if(pos == -1)
			System.out.println("Element not found");
		else
			System.out.println("Element is found at position: " +
pos);
	}


	public static void main (String[] args)
	{
		Stack<Integer> stack = new Stack<Integer>();

		stack_push(stack);
```

```
            stack_pop(stack);
            stack_push(stack);
            stack_peek(stack);
            stack_search(stack, 2);
            stack_search(stack, 6);
      }
}

// 23.queue in java

import java.util.*;
class Book implements Comparable<Book>{
int id;
String name,author,publisher;
int quantity;
public Book(int id, String name, String author, String publisher, int
quantity) {
    this.id = id;
    this.name = name;
    this.author = author;
    this.publisher = publisher;
    this.quantity = quantity;
}
public int compareTo(Book b) {
    if(id>b.id){
        return 1;
    }else if(id<b.id){
        return -1;
    }else{
    return 0;
    }
}
}
public class LinkedListExample {
public static void main(String[] args) {
    Queue<Book> queue=new PriorityQueue<Book>();
    //Creating Books
    Book b1=new Book(121,"Let us C","Yashwant Kanetkar","BPB",8);
    Book b2=new Book(233,"Operating System","Galvin","Wiley",6);
    Book b3=new Book(101,"Data Communications &
Networking","Forouzan","Mc Graw Hill",4);
    //Adding Books to the queue
    queue.add(b1);
    queue.add(b2);
    queue.add(b3);
    System.out.println("Traversing the queue elements:");
    //Traversing queue elements
    for(Book b:queue){
    System.out.println(b.id+" "+b.name+" "+b.author+" "+b.publisher+"
"+b.quantity);
    }
    queue.remove();
    System.out.println("After removing one book record:");
    for(Book b:queue){
        System.out.println(b.id+" "+b.name+" "+b.author+" "+b.publisher+"
"+b.quantity);
        }
}
```

```
}
```

TheEnd