Assignment 1 : 4 January 2018

Name : Divya Garg

Employee ID : 33112

**Q 1 :  Can abstract class have constructors in Java?**

Ans : Yes , Abstract class can have constructor . Child class of Abstract class will implicitly call the default constructor constructor of parent class and to call the parameterized constructor of parent class , we will use super keyword.

class Father {

Father(String name){

System.out.println("Father;s constructor : "+name);

}

}

class Son extends Father{

Son(String name){

super("Ram");

System.out.println("Son's constructor : "+name);

}

}

public class Q1 {

public static void main(String[] args) {

Son son=new Son("Luv");

}

}

**Q2 : Can abstract class implements interface in Java? do they require to implement all methods?**

Ans : Abstract class can implements interface in java . For an abstract class , it is not mandatory to implement all methods . This is called Partial implementation. Abstract class have the choice of implementing zero or more than one method .

interface MyInterface {

void show();

void fun();

}

abstract class MyAbstractclass implements MyInterface {

public void show(){

System.out.println("I m show");

}

}

class MyConcrete extends MyAbstractclass {

public void fun(){

System.out.println("I m fun");

}

}

public class Q2 {

public static void main(String[] args) {

MyConcrete obj=new MyConcrete();

obj.fun();

obj.show();

}

}

**Q 3 : Can abstract class be final in Java?**

Ans : No , Abstract class cant be final in Java .Abstract class needs to be inherited whereas final class cant be inherited , so Abstract and final used together is a illegal combination , they are contradicting each other.

**Q 4 : Can abstract class have static methods in Java?**

Ans : Yes , Abstract class can have static methods but abstract methods cant be static. Abstract and static used together is a illegal combination.

**Q 5 : Can you create instance of abstract class?**

Ans : No , we cant create instance of abstract class . We can only create the reference variable of an abstract class and store the object of its child class.

abstract class Animal {

abstract void sound();

}

class Cat extends Animal{

void sound(){

System.out.println("Mew Mew");

}

}

public class Q5 {

public static void main(String[] args) {

Animal obj=new Cat();

obj.sound();

}

}

**Q 6 : Is it necessary for abstract class to have abstract method?**

Ans : No , it is not necessary for an abstract class to have abstract methods.We can have abstract class without any abstract method.e.g. In java.awt.event package , MouseAdapter is a class which is not having any abstract method. In Java APIs , we do have several classes of this type and we can also create User-defined classes in same manner.

**Q 7 : Difference between abstract class and interface in Java?**

Ans :

|  |  |
| --- | --- |
| Abstract class | Interface |
| Abstract class can have abstract as well as abstract methods. | Interface can have only abstract methods , infact all the methods of interface are by default public and abstract |
| Variables of Abstract class will be in the manner in which you create them | All the variables of interface are by default public static and final |
| Any class can inherit only one abstract class at a time. | Any class can implement one or more than one interface at a time. |
| We can create constructors in abstract class | We cant create constructors for interface |
| Extends keyword is used to inherit any abstract class | Implements keyword is used to inherit any interface. |

**Q 8 : When do you favor abstract class and interface in java?**

Ans :-

|  |  |
| --- | --- |
| Abstract class | Interface |
| Inheriting abstract class represents (is-a) relationship. | Implementing interface represent can do relationships |
| Use abstract class if you have a default implementation of some behavior that child classes do not have to implement. | If there is no default or common behavior among all the clases that are inheriting from abstract class then interface may be a better choice. |
| Prefer abstract classes if your contract has a possibility of changing over time . So if you are using an abstract class and need to add a new method to your abstract class , you can happily add that without breaking any code using that class. | The same is not true for interfaces. |
| Since multiple inheritance is not possible in java , you cannot inherit your class from two abstract classes. | You can implement more than one interface at a time. |

To elaborate first point,let’s consider following two classes :-

public class Camera {

public void shoot(){ //code }

}

public class Gun {

public void shoot(){ //code }

}

Both Camera and Gun can shoot , that is their capability . But they are both not the same type of things , they are completely different .So an interface like below would make more sense here :-

public interface IShootable {

void shoot();

}

public class Camera implements IShootable {

public void shoot(){

//Take a picture here

}

}

public class Gun implements IShootable {

public void shoot(){

//Hit the target

}

}

e.g. 2 public abstract class Shape {

void draw();

}

public class Rectangle extends Shape {

public void draw(){

//draw a rectangle here

}

}

public class Circle extends Shape {

public void draw(){

//Draw a circle here

}

}

Rectangle and Circle inheriting from Shape makes perfect sense here because Rectangle/Circle are a type of Shape.

**Q 9 :  What is abstract method in Java?**

Ans : Abstract method is the method which is only having the declaration , not having any definition . In java , abstract access modifier is used to declare any abstract method. Any class having any abstract method needs to be declared as abstract . Abstract method cant be declared as static and final.

abstract class Animal {

abstract void sound();

}

**Q 10 : Can abstract class contains main method in Java ?**

Ans : Abstract class can have main() method .

public abstract class Q10 {

public static void main(String[] args){

System.out.println("Main of Abstract class");

}

}

**Q 11 :   what is static block in java?**

Ans : Static block is used for initializing the static variables.This block gets executed when the class is loaded in the memory. A class can have multiple Static blocks, which will execute in the same sequence in which they have been written into the program. Static block is the block of code which gets executed before the main method . In Java program , when any class will gets loaded into the memory , first the static variables will get initialized , then static blocks will get executed and then main method will get invoked.

public class StaticBlockDemo {

static int num;

static String mystr;

//First static block

static {

System.out.println("Static Block 1");

num=68;

mystr="Block1";

}

//second static block

static {

System.out.println("Static Block 2");

num=98;

mystr="Block2";

}

public static void main(String[] args) {

System.out.println("Value of num = "+num);

System.out.println("Value of mystr= "+mystr);

}

}

**Q 12 :   What is the need of static block?**

* Suppose if we have any static member which is having complex initialization , cant be declared in one line so static block can be used to initialize such type of static variables. Suppose , u have a collection that you want to initialize and add 10 elements , this can be done in static block.
* If we have some code that we want to do only once in the complete lifecycle of the object , that task can also be done inside the static block.

**Q 13 : Can we overload static methods in java?**

Ans : Yes , we can overload static methods in java. In Method overloading , whether method is static or non-static doesn’t make any difference.

// filename Test.java

public class Test {

    public static void foo() {

        System.out.println("Test.foo() called ");

    }

    public static void foo(int a) {

        System.out.println("Test.foo(int) called ");

    }

    public static void main(String args[])

    {

        Test.foo();

        Test.foo(10);

    }

}

**Q 14 : Can we call super class static methods from sub class?**

Ans : Yes , we can super class static methods from sub class .

**Q 15 : What is the difference between final and static keywords?**

Ans :-

final :- final is a keyword which can be used with class , methods , variables

Final class – Any class declared using final keyword cant be inherited . For example java.lang.String class is final class in java , it cant be inherited.

Final method : - Methods declared using final keyword cant be overridden.

Final variable :- final keyword is used to make constant in java .

Static :- static keyword can be used with variable , method and we also have static block in java . Any variable or method declared using static keyword is the property of Class. It can be accessed with ClassName eg.ClassName.variableName or ClassName.methodName() . All the objects share the single copy of static variable and methods.

**Q 16 : Write a note on covariant return type with example code.**

Ans : Before JDK 1.5 , it was not possible to override a method by changing the return type. When we override a parent class method , the name, argument types and return type of the overriding method in child class has to be exactly same as that of parent class method. Overriding method was said to be invariant with respect to return type. But from JDK 1.5 , it is possible to have different return type for overriding method in child class , but child’s return type should be sub-type of parent’s return type. Overriding method becomes variant with respect to return type.

class A {}

class B extends A {}

class Base {

A fun(){

System.out.println("Base's fun");

return new A();

}

}

class Derived extends Base {

B fun(){

System.out.println("Derived's fun");

return new B();

}

}

public class Q16\_CovariantReturnType {

public static void main(String[] args){

Derived obj=new Derived();

obj.fun();

}

}

**Q 17 : Write a note on Enum with example code.**

Ans : Enum in java is a data type that contains fixed set of constants. It can be used for days of the week (SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATURDAY) , directions (North,South,East and West) etc. The java enum constants are static and final implicitly . It is available from JDK 1.5.

Java Enums can be thought of as classes that have fixed set of constants.

Points to remember for Java Enum :-

1. Enum improves type safety
2. Enum can be easily used in switch
3. Enum can be traversed
4. Enum can have fields , constructors and methods.
5. Enum may implement many interfaces but cannot extend any class because it internally extends Enum class.
6. Enum can be declared inside the class as well as outside the class.

//enum Season { WINTER,SPRING,SUMMER,FALL}

public class EnumExample1 {

public enum Season { WINTER,SPRING,SUMMER,FALL}

//values() method returns an array containing all the values of the Enum

public static void main(String[] args){

for(Season s:Season.values()){

System.out.println(s);

}

}

}

E.g. 2

enum Season {WINTER,SPRING,SUMMER,FALL}

public class EnumExample2 {

public static void main(String[] args){

Season s=Season.WINTER;

System.out.println(s);

}

}

e.g. 3

public class EnumExample3 {

enum Season { WINTER(5),SPRING(10),SUMMER(15),FALL(20);

private int value;

private Season(int value){

this.value=value;

}

}

public static void main(String[] args){

for(Season s:Season.values()){

System.out.println(s+" "+s.value);

}

}

}

//constructor of enum type is private . If you don’t declare private compiler internally creates private constructor.

e.g. 4 – Example of Applying enum on switch statement –

public class EnumExample4 {

enum Day{ SUNDAY , MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATURDAY}

public static void main(String[] args){

Day day=Day.MONDAY;

switch(day){

case SUNDAY:

System.out.println("Sunday");

break;

case MONDAY:

System.out.println("Monday");

break;

default : System.out.println("Other Day");

}

}

}

**Q 18 : Write a note on use of super keyword and super() method.**

Ans : super keyword has several uses which are as follows :-

1. Super() is used to call the parameterized constructor of super class from the constructor of sub class. Call to super() must be the first statement in the constructor.

class Area {

int length,breadth;

Area(int length,int breadth){

this.length=length;

this.breadth=breadth;

}

public int area(){

return length\*breadth;

}

}

class Volume extends Area {

int height;

Volume(int l,int b,int h){

super(l,b);

height=h;

}

public int volume(){

return length\*breadth\*height;

}

}

public class SuperDemo {

public static void main(String[] args) {

Volume obj=new Volume(2,3,4);

System.out.println("Area : "+obj.area());

System.out.println("Volume : "+obj.volume());

}

}

2. When super class and sub class both are having a variable with same name , then sub class variable will hide the super class variable , then to access super class variable from sub class , we can use super keyword.

1. In case of Method overriding , If we want to call super class method from sub class then we can use super keyword

Note : Super keyword cant be used in static method and in static block.

class Parent {

int x=20;

void display(){

System.out.println("I m display of fun");

}

}

class Child extends Parent {

int x=30;

void display(){

super.display();

System.out.println("I m display of child : "+x+" "+super.x);

}

}

public class SuperDemo\_2 {

public static void main(String[] args){

Child obj=new Child();

obj.display();

}

}

**Q 19 : Write a code to implement abstraction using interface.**

interface Playable {

void play();

}

class Veena implements Playable{

public void play(){

System.out.println("Playing Veena");

}

}

class Saxophone implements Playable{

public void play(){

System.out.println("Playing Saxophone");

}

}

public class InterfaceMain {

public static void main(String[] args) {

Playable p1=new Veena();

Playable p2=new Saxophone();

p1.play();

p2.play();

}

}

**Q 20 : Write a Java program to sort a numeric array and a string array.**

//Program sorting numeric array

public class Program1 {

public static void main(String[] args) {

int[] arr={85,15,96,5,100,5,20,-8};

int temp=0;

System.out.println("\n=====Elements Before Sort======");

for(int i=0;i<arr.length;i++){

System.out.println(arr[i]);

}

//code to sort numeric array...

for(int i=0;i<arr.length;i++){

for(int j=i+1;j<arr.length;j++){

if(arr[i]>arr[j]){

temp=arr[i];

arr[i]=arr[j];

arr[j]=temp;

}

}

}//end loop

System.out.println("\n=====Elements After Sort======");

for(int i=0;i<arr.length;i++){

System.out.println(arr[i]);

}

}

}

//Program sorting String Array

public class Program2 {

public static void main(String[] args) {

String[] strArray={"Divya","Komilla","Anjali","Ashutosh","Rajesh","Neha","Najir"};

String temp="";

System.out.println("\n====Array Before Sort======");

for(String str:strArray){

System.out.println(str);

}

//logic to sort String Array....

for(int i=0;i<strArray.length;i++){

for(int j=i+1;j<strArray.length;j++){

if((strArray[i].compareTo(strArray[j]))>1){

temp=strArray[i];

strArray[i]=strArray[j];

strArray[j]=temp;

}

}

}

System.out.println("\n====Array After Sort======");

for(String str:strArray){

System.out.println(str);

}

}

}

Alternate to this , Arrays.sort() method can also be used for Sorting Array.

**Q 21 : Write a Java program to sum values of an array.**

Ans :

public class Program3 {

public static void main(String[] args) {

int[] arr={10,8,9,52,36};

int sum=0;

for(int i=0;i<arr.length;i++){

sum=sum+arr[i];

}

System.out.println("Sum of Elements of an Array : "+sum);

}

}

**Q 22 : Write a Java program to remove a specific element from an array.**

Ans :

public class Program4 {

public static void main(String[] args) {

int[] arr={85,69,5,369,1,11};

int[] newArr=new int[arr.length-1];

Scanner sc=new Scanner(System.in);

System.out.println("Enter the element that u want to search");

int numToSearch=sc.nextInt();

System.out.println("====Elements Before removing element=====");

for(int ii:arr){

System.out.println(ii);

}

int j=0;

for(int i=0;i<arr.length-1;i++){

if(arr[i]==numToSearch){

newArr[i]=arr[i+1];

j=1;

}

else {

if(j==1){

newArr[i]=arr[i+1];

}

else {

newArr[i]=arr[i];

}

}

}//end loop

System.out.println("====Elements After removing element=====");

for(int ii:newArr){

System.out.println(ii);

}

}

}

**Q 23 : Write a Java program to reverse an array of integer values.**

public class Program5 {

public static void main(String[] args) {

int[] arr={85,69,5,369,1,11};

System.out.println("\n=====Array Before Reverse=====");

for(int i:arr){

System.out.println(i);

}

int temp=0;

//logic to reverse an array...

for(int i=0,j=arr.length-1;i<j;i++,j--){

temp=arr[i];

arr[i]=arr[j];

arr[j]=temp;

}

System.out.println("\n=====Array After Reverse=====");

for(int i:arr){

System.out.println(i);

}

}

}

**Q 24 : Write a Java program to find the duplicate values of an array of integer values.**

Ans :

public class Program6 {

public static void main(String[] args) {

int[] arr={85,11,5,369,5,11};

int dup=arr[0];

for(int i=0;i<arr.length;i++){

for(int j=i+1;j<arr.length;j++){

if(arr[i]==arr[j]){

System.out.println("Duplicate element : "+arr[i]);

}

}

}//end loop

}

}