**Core Java Assignments**

1. Write a program to print factorial of N ( without using any loop)

import java.lang.Math;

import java.util.Scanner;

// headers MUST be above the first class

// one class needs to have a main() method

public class HelloWorld

{

// arguments are passed using the text field below this editor

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

int i,fact=1;

System.out.println("Enter a number");

int number=sc.nextInt();

fact = factorial(number);

System.out.println("Factorial of "+number+" is: "+fact);

}

static int factorial(int n){

if (n == 0)

return 1;

else

return(n \* factorial(n-1));

}

}

1. There is an animal class which has the common characteristics of all animals. Dog, Horse, Cat are animals(sub-class). Each can shout, but each shout is different. Use polymorphism to create objects of same and using an animal variable, make each of the animals shout.

public class Polymorphism {

public static void main(String[] args){

Animal obj1 = new Dog();

Animal obj2 = new Cat();

Animal obj3 = new Horse();

obj1.shout(); //output is bark..

obj2.shout(); //output is bark..

obj3.shout();

}

}

class Animal{

public void shout(){

System.out.println("Parent animal's shout");

}

}

class Dog extends Animal{

public void shout(){

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

}

}

class Cat extends Animal{

public void shout(){

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

}

}

class Horse extends Animal{

public void shout(){

System.out.println("neigh");

}

}

1. Create an abstract class Instrument which is having the abstract function play. Create three more sub classes from Instrument which is Piano, Flute, Guitar. Override the play method inside all three classes printing a message

“Piano is playing tan tan tan tan ” for Piano class

“Flute is playing toot toot toot toot” for Flute class

“Guitar is playing tin tin tin ” for Guitar class

Create an array of 10 Instruments.

Assign different type of instrument to Instrument reference.

Check for the polymorphic behavior of play method.

abstract class instrument

{

abstract public void play();

}

class piano extends instrument

{

public void play()

{

System.out.println("Piano is playing tan tan tan tan");

}

}

class flute extends instrument

{

public void play()

{

System.out.println("Flute is playing toot toot toot toot");

}

}

class guitar extends instrument

{

public void play()

{

System.out.println("Guitar is playing tin tin tin tin");

}

}

public class inst

{

public static void main(String [] args)

{

instrument ins[]=new instrument[10];

for(int i=0;i<10;i++)

{

if(i==1 || i==5 || i==9)

ins[i]=new piano();

else if(i==3 || i==4 || i==7)

ins[i]=new flute();

else

ins[i]=new guitar();

ins[i].play();

if(ins[i] instanceof piano)

System.out.println("InstanceOf Piano");

else if(ins[i] instanceof flute)

System.out.println("InstanceOf Flute");

else System.out.println("InstanceOf Guitar");

System.out.println();

}

}

}

1. Write an interface called Playable, with a method void play();

Let this interface be placed in a package called music.Write a class called Veena which implements Playable interface. Let this class be placed in a package music.string.Write a class called Saxophone which implements Playable interface. Let this class be placed in a package music.wind.Write another class Test in a package called live. Then,

a. Create an instance of Veena and call play() method

b. Create an instance of Saxophone and call play() method

c. Place the above instances in a variable of type Playable and then call play().

package c08.music5;

import com.bruceeckel.simpletest.\*;

import c07.music.Note;

interface Instrument {

// Compile-time constant:

int I = 5; // static & final

// Cannot have method definitions:

void play(Note n); // Automatically public

String what();

void adjust();

}

class Veena implements Instrument {

public void play(Note n) {

System.out.println("Veena.play() " + n);

}

public String what() { return "Veena"; }

public void adjust() {}

}

class Saxophone implements Instrument {

public void play(Note n) {

System.out.println("Saxophone.play() " + n);

}

public String what() { return "saxophone"; }

public void adjust() {}

}

public class Music5 {

private static Test monitor = new Test();

// Doesn't care about type, so new types

// added to the system still work right:

static void tune(Instrument i) {

// ...

i.play(Note.MIDDLE\_C);

}

static void tuneAll(Instrument[] e) {

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

tune(e[i]);

}

public static void main(String[] args) {

// Upcasting during addition to the array:

Instrument[] orchestra = {

new Veena(),

new Saxophone(),

};

tuneAll(orchestra);

monitor.expect(new String[] {

"Veena.play() Middle C",

"Saxophone.play() Middle C",

});

}

}

1. Write a program to accept name and age of a person from the command prompt(passed as arguments when you execute the class) and ensure that the age entered is >=18 and < 60. Display proper error messages. The program must exit gracefully after displaying the error message in case the arguments passed are not proper. (Hint : Create a user defined exception class for handling errors.)

class InvalidAge Exception extends Exception

{

public InvalidProductException(String s)

{

super(s);

}

}

Public class Person

{

Public static void main(String args[])

If(args.length>0)

{

try{

String Name =args[0];

int age = Integer.parseInt(args[1]);

if(age<18 && age >60)

throw new InvalidAgeException("Age is Invalid");

else

System.out.println("Name:"+Name+" Age:"+age);

}

}

}

1. Write a program to assign the current thread to t1. Change the name of the thread to MyThread. Display the changed name of the thread. Also it should display the current time. Put the thread to sleep for 10 seconds and display the time again.

class Multi extends CurrentThread{

public void run(){

System.out.println("thread is running...");

try {

while (true) {

java.util.Date d = new java.util.Date();

SimpleDateFormat sdf = new SimpleDateFormat("hh:MM:ss a");

jLabel1.setText(sdf.format(new Date(System.currentTimeMillis())));

Thread.sleep(1000);

}

} catch (Exception e) {

}

}

public static void main(String args[]){

Thread t1=new Thread();

t1.start();

System.out.println("Name of t1:"+t1.getName());

  t1.setName("MyThread");

  System.out.println("After changing name of t1:"+t1.getName());

 }

}

1. Create an ArrayList of Employee( id,name,address,sal) objects and search for particular Employee object based on id number and name.

public class emp {

int id ;

String name;

String address;

int salary;

public emp(int i, String name, String add, int sal)

{

super();

this.id = i;

this.name = name;

this.address=add;

this.salary=sal;

}

}

public class Test {

public static void main(String[] args)

{

int givenEmpId = 3;

String givenEmpName = "Pinky";

ArrayList<emp> empList = new ArrayList<emp>();

empList.add(new emp(1,"Priya","Hyderabad",10000));

empList.add(new emp(2,"Vijaya","Banglore",20000));

empList.add(new emp(3,"Pinky","Mysore",30000));

for ( emp currEmp : empList)

{

if(currEmp.id==givenEmpId && currEmp.name==givenEmpName)

{

System.out.println("Employee details are===>>"+"\nEmployee id is:"+currEmp.id+"\nEmployee name is:"+currEmp.name+"\nEmployee address is:"+currEmp.address+"\nEmployee salary is:"+currEmp.salary);

}

}

}

}