1. Why OOPS and why it is important?

Ans: OOPS is programming technique that employs objects rather than just functions and procedure.

All objects are grouped into classes in object oriented programming.

OOPS integrates real world concepts into programming such as Inheritance, Polymorphism and Abstraction.

OOPS Allows more clarity in programming thereby allowing simplicity in solving complex problems.

OOPS Reduces redundancy

OOPS Provides ability to bind both data and Code together

Allows in keeping sensitive data confidential.

1. What are classes and objects?

Ans: A class is a blue print of an Object. It is a data type that use specifies.

1. Compare inheritance and polymorphism

Ans: With inheritance a derived class inherits the already existing class’s feature. Whereas polymorphism allows class methods to exist in multiple forms.

1. What is encapsulation?

Ans: Encapsulation is process of binding data members and methods together to a specific job without revealing unnecessary details.

1. Can we instantiate an abstract class? Ans: No
2. How an abstract class is different from Interfaces?

Ans: Both contain only method declaration and not their implementation. When an interface is implemented, the subclass must define all its methods and provide its implementation. When an abstract class in inherited, the subclass does not need to provide the definition of its abstract method, until and unless the subclass is using it.

1. What is the fundamental difference between abstraction and encapsulation?

Ans: abstraction is about expressing external simplicity and encapsulation is about hiding internal complexity.

1. Both constructor and destructor cannot be overloaded in its subclass. The composition is way to implement the has a relationship. The copy constructor and the assignment operator both are used to initialize one object using another object.
2. Overloading is the concept of OOP which employes two or more methods in the class with same name and different method signature. Which is also known as runtime polymorphism
3. //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unit price 5.00, quantity 10, name John , find amount to be paid which is unitPrice \* quantity

write a test class

Class Customers {

private String name;

private float unitPrice;

private int quantity;

// default constructor

public Customers() {

}

// parameterized constructor

public Customers(string name, float unitPrice, int quantity) {

this.name = name;

this.unitPrice = unitPrice;

this.quantity = quantity;

}

// setter

public void setName(string name){

this.name = name;

}

// getter

public string getName() {

return this.name;

}

public float getAmountToBePaid() {

return this.unitPrice \* this.quantity;

}

}

class Test {

public static void main(String[] args) {

ArrayList<Customers> arr = new ArrayList<Customers>();

Customers cust1 = new Customers(“John”, 5.00, 10);

Customers cust2= new Customers(“xyz”, 4.00, 10);

System.out.println(cust.getAmountToBePaid());

System.out.println(cust.getName());

// add customers into array

arr.add(cust1);

arr.add(cust2);

// find sum

float sum = 0;

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

sum += arr.get(i).getAmountToBePaid();

}

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

static int isIndexofOfNonDuplicate(int[] a) {

int counter = 0;

int index=0;

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

int count = 0;

int min=0;

for(int j=0; j < a.length; j++) {

if(a[i] == a[j])

count ++;

if(j == a.length-1) {

if(count == 1)

{

if (counter==0)

{

min=a[i] ;

counter++;

index = i;

}

if (a[i]< min)

{

min=a[i];

index = i;

}

}

}

}

}

return index;

}

Public static int fib(int n)

{ if (n<=1)

Return n;

Else

Return fib(n-1)+fib(n-2);

}

Public static int power( int base, int powerRaised)

{

If (powerRaised==0)

Return 1;

Else

Return base\*power(base, powerRaised-1);

}

class FibonacciExample1{

public static void main(String args[])

{

int n1=0,n2=1,n3,i,count=10;

System.out.print(n1+" "+n2);//printing 0 and 1

for(i=2;i<count;++i)//loop starts from 2 because 0 and 1 are already printed

{

n3=n1+n2;

System.out.print(" "+n3);

n1=n2;

n2=n3;

}

}}

https://javarevisited.blogspot.com/2017/07/top-50-java-programs-from-coding-Interviews.html#axzz7wCMYfDpx