**Question #1**

**(a)**

**public static <T, S> void func(T a1, S a2, T a3){**

**System.*out*.println(a1 + ", "+a2+ ", "+ a3);**

**}**

**(b)**

In java multiple inheritance is not allowed but multiple inheritance of interfaces can be done: here is the example:

interface A {

int *a* = **1;**

}

interface B extends A{

int *a* = **10;**

}

interface C extends A{

int *a* = **20;**

}

public class Q1\_b implements B**,** C {

public static void main(String[] args) {

System.*out*.println(a)**;**

}

}

Here interface B and C extends A and have the same variable and when a class is implemented both B and C diamond problems occur because the parent is the same and the variable name conflicts and the compiler gives a compile time error. To solve this we need to give reference to the variable.

**(C)**

No virtual functions in java so giving difference between abstract functions and normal function as sir said:

Abstract functions are functions without anybody, if there is an abstract function in the class then the whole class need to be declared abstract.

**(D)**

**IS-A relationship:**This is the relationship when a class is inherited from a parent, the child class has the properties of the parent class but can also have its own properties, eg: there is a class named person and we extend teacher from that class, now this is an IS A relation. Teacher **IS A** person

**Has-A relationship:**

When a associated occurs then there is HAS-A relationship, eg. object Engine is initialized in the class of Car, now we can say that Car **HAS A** engine, this is just simple composition.

**(E)**

No, write the whole code (that can have exceptions) in the try block and then write different catch blocks according to the possible exceptions.

**Question #2**

**(A)**

No ofsteam class in java, Java have different mode of streams in java, like fileInputStream, fileOutputStream, but no default mode of opening in java, we have to make separate objects for writing and reading data from the file.

If the question is asking about the appends and all that stuff then in FileOutoutStream we can describe in boolean that we want to append or not.

**(B)**

Functions overriding / polymorphism

**(C)**

Abstraction

**(D)**

idk EOF maybe but in while loop we check in the string read is not null

**(E)**

Try catch

**(F)**

Generic

**(G)**

Overridden functions

**(H)**

Try catch

**(I)**

False

**(J)**

finalize()

**Question #3**

**(A)**

According to the java (UNICODE) output is:  
16

24

**(B)**

No sizeof operator to measure object size in java.

But phir bhi tukka:

16 character in string and 2 bytes for character in unicode the 32 is for string only, then we need to calculate for the class, it is definitely more then 32

**(C)**

No pointers in java, IndexOutOfBound for java users lol.

**(D)**

I program in java, cpp isIndexOutOfBound for me

**Error handling :: Val: 0 Ex: 10**

**All Exception Catch: val: 0**

According to chatGPT

**(E)**

10 according to chatGPT

**Question #4**

**(a)**

**// Default constructor**

**public Tournament(){**

**sport = new Sport();**

**}**

**// overloaded constructor which receives attributes for sports in parameters**

**public Tournament(String current\_champ, int start\_year, String headquarter\_location){**

**sport = new Sport(current\_champ, start\_year, headquarter\_location);**

**}**

**(B)**

public void begin\_Tournament(Sport s){

if (s.headquarter\_location.equalsIgnoreCase("karachi") || s.headquarter\_location.equalsIgnoreCase("toronto")){

System.*out*.println("Tournaments begins")**;**

}

else{

System.*out*.println("Location not valid")**;**

}

}

**(C)**

**class WorldCup extends Tournament{**

**public void begin\_Tournament(Sport s){**

**if (!(s.headquarter\_location.equalsIgnoreCase("karachi") || s.headquarter\_location.equalsIgnoreCase("toronto")) && s.start\_year > 1950){**

**System.*out*.println("Tournament begins");**

**}**

**}**

**}**

**(D)**

**public <T> void printIt(T value){**

**System.*out*.println(value);**

**}**

**Question #5**

import java.util.Scanner**;**

class overflow extends Exception{

public overflow(){

System.*out*.println("Count Overflow")**;;**

}

}

class underflow extends Exception{

public underflow(){

System.*out*.println("Count Underflow")**;**

}

}

class Jumping\_Castle {

int count**;**

public Jumping\_Castle(){

this.count = **0;**

}

public void Enter(){

try{

if (count==**10**) {

throw new overflow()**;**

}

else{

count++**;**

System.*out*.println("Entered successfully!")**;**

System.*out*.println("Count: " + this.count)**;**

}

} catch (overflow e){

e.getMessage()**;**

}

}

public void Exit(){

try{

if (count == **0**){

throw new underflow()**;**

}

else{

count--**;**

System.*out*.println("Exit successfully")**;**

System.*out*.println("Count: " + this.count)**;**

}

} catch(underflow e) {

System.*out*.println(e.getMessage())**;**

}

}

public void display(){

System.*out*.println("Count : " + this.count)**;**

}

}

public class Q5{

public static void main(String[] args) {

int ch=**0;**

Scanner sc = new Scanner(System.*in*)**;**

Jumping\_Castle jm = new Jumping\_Castle()**;**

while (ch != **4**){

System.*out*.println("1. Enter\n2.Exit\n3.Show count\n4.exit program: ")**;**

ch=sc.nextInt()**;**

switch(ch){

case (**1**)->{

jm.Enter()**;**

}

case (**2**) -> {

jm.Exit()**;**

}

case (**3**) -> {

jm.display()**;**

}

case (**4**) -> {

System.*exit*(**0**)**;**

}default -> {

System.*out*.println("invalid input!")**;**

}

}

}

}

}

**Question #6**

abstract class FILE{

int size**;**

int location**;**

String dateCreated**;**

String modifiedDate**;**

public FILE(int size**,** int location**,** String dateCreated**,** String modifiedDate) {

this.size = size**;**

this.location = location**;**

this.dateCreated = dateCreated**;**

this.modifiedDate = modifiedDate**;**

}

abstract void open()**;**

abstract void print()**;**

}

class PDF extends FILE{

public PDF(int size**,** int location**,** String dateCreated**,** String modifiedDate) {

super(size**,** location**,** dateCreated**,** modifiedDate)**;**

}

void open(){

System.*out*.println("Opening PDF file")**;**

}

void print(){

System.*out*.println("Printing PDF file")**;**

}

}

class ASCII extends FILE{

public ASCII(int size**,** int location**,** String dateCreated**,** String modifiedDate) {

super(size**,** location**,** dateCreated**,** modifiedDate)**;**

}

void open(){

System.*out*.println("Opening ASCII file")**;**

}

void print(){

System.*out*.println("Printing ASCII file")**;**

}

}

class PS extends FILE{

public PS(int size**,** int location**,** String dateCreated**,** String modifiedDate) {

super(size**,** location**,** dateCreated**,** modifiedDate)**;**

}

void open(){

System.*out*.println("Opening PS file")**;**

}

void print(){

System.*out*.println("Printing PS file")**;**

}

}

public class Q6 {

public static void main(String[] args) {

PDF p1 = new PDF(**12, 19023,** "12/2/2022"**,** "20/5/2023")**;**

ASCII a1 = new ASCII(**12, 54321,** "23/4/2018"**,** "2/6/2020")**;**

PS ps1 = new PS(**12, 19023,** "2/8/2018"**,** "09/1/2020")**;**

p1.print()**;**

a1.print()**;**

ps1.print()**;**

System.*out*.println("p1 and ps1 same?: " + *compare*(p1**,** ps1))**;**

System.*out*.println("a1 and p1 same?: " + *compare*(a1**,** p1))**;**

}

public static boolean compare(FILE f1**,** FILE f2){

if ((f1.size == f2.size) && (f1.location == f2.location)){

return true**;**

}

else {

return false**;**

}

}

}

**Question #7**

**import javax.swing.plaf.metal.MetalIconFactory;**

**import java.io.\*;**

**class laptops implements Serializable {**

**private int id;**

**private float price;**

**private int hdd;**

**private int ram;**

**public laptops(int id, float price, int hdd, int ram) {**

**this.id = id;**

**this.price = price;**

**this.hdd = hdd;**

**this.ram = ram;**

**}**

**public int getId() {**

**return id;**

**}**

**public void setId(int id) {**

**this.id = id;**

**}**

**public float getPrice() {**

**return price;**

**}**

**public void setPrice(float price) {**

**this.price = price;**

**}**

**public int getHdd() {**

**return hdd;**

**}**

**public void setHdd(int hdd) {**

**this.hdd = hdd;**

**}**

**public int getRam() {**

**return ram;**

**}**

**public void setRam(int ram) {**

**this.ram = ram;**

**}**

**public void print(){**

**System.*out*.println("ID: " + this.id);**

**System.*out*.println("Price: " + this.price);**

**System.*out*.println("HDD: " + this.hdd);**

**System.*out*.println("Ram: " + this.ram);**

**}**

**}**

**class HP extends laptops{**

**private int portsCount;**

**public HP(int id, float price, int hdd, int ram, int portsCount) {**

**super(id, price, hdd, ram);**

**this.portsCount = portsCount;**

**}**

**public void print(){**

**super.print();**

**System.*out*.println("Ports: " + this.portsCount);**

**}**

**}**

**class Dell extends laptops{**

**private float screenSize;**

**public Dell(int id, float price, int hdd, int ram, float screenSize) {**

**super(id, price, hdd, ram);**

**this.screenSize = screenSize;**

**}**

**public void print(){**

**super.print();**

**System.*out*.println("Screen size: " + this.screenSize);**

**}**

**}**

**class Asus extends laptops{**

**private int graphicCardSize;**

**public Asus(int id, float price, int hdd, int ram, int graphicCardSize) {**

**super(id, price, hdd, ram);**

**this.graphicCardSize = graphicCardSize;**

**}**

**public void print(){**

**super.print();**

**System.*out*.println("Graphic card size: " + this.graphicCardSize);**

**}**

**}**

**public class Q7 {**

**public static void main(String[] args) throws IOException {**

**HP hp = new HP(12, 12000, 512, 12, 4);**

**HP hp2 = new HP(17, 478334, 512, 12, 4);**

**Dell dell = new Dell(13, 90000, 256, 12, 12);**

**Asus asus = new Asus(14, 140000, 1024, 3, 13);**

**File f = new File("datafile.bin");**

**if (!f.exists()){**

**try {**

**System.*out*.println("File Doesn't exist: ");**

**System.*out*.println("Writing objects: ");**

**ObjectOutputStream ois = new ObjectOutputStream(new FileOutputStream(f));**

**ois.writeObject(hp);**

**ois.writeObject(asus);**

**ois.writeObject(dell);**

**ois.writeObject(hp2);**

**ois.close();**

**} catch (IOException e){**

**System.*out*.println(e.getMessage());**

**}**

**}**

**else {**

**float hpPrice=0;**

**float dellPrice=0;**

**float asusprice=0;**

**HP h1 = null;**

**Dell d1 = null;**

**Asus a1 = null;**

**try {**

**ObjectInputStream oos = new ObjectInputStream(new FileInputStream(f));**

**Object obj;**

**while(true){**

**try{**

**obj = oos.readObject();**

**if (obj instanceof HP){**

**h1 = (HP) obj;**

**h1.print();**

**hpPrice+=h1.getPrice();**

**}**

**else if (obj instanceof Dell){**

**d1 = (Dell) obj;**

**d1.print();**

**dellPrice+=d1.getPrice();**

**}**

**else if (obj instanceof Asus){**

**a1 = (Asus) obj;**

**a1.print();**

**asusprice+=a1.getPrice();**

**}**

**} catch (EOFException e) {**

**break;**

**}**

**}**

**File f2 = new File("report.txt");**

**BufferedWriter bw = new BufferedWriter(new FileWriter(f2));**

**bw.write("HP: " + hpPrice + "\n");**

**bw.write("DELL: " + dellPrice+ "\n");**

**bw.write("ASUS: " + asusprice+ "\n");**

**bw.write("===================="+ "\n");**

**bw.write("Total: " + (hpPrice+dellPrice+asusprice) + "\n");**

**bw.close();**

**oos.close();**

**} catch (IOException e){**

**System.*out*.println(e.getMessage());**

**} catch (ClassNotFoundException e) {**

**System.*out*.println("CCCC");**

**}**

**}**

**}**

**}**

**Here the programs first checks if the file exist or not, if not then it creates the file first, if the file exists then if loops and read all the objects and check what type of object then sums and prints the object, finally if creates a file called report and write the data there.**