1.Why do we need to handle the exception? And how do we handle the exception? Explain with example

Exception is an error event that can happen during the execution of program and disrupts the flow of program,java provides robust and object oriented way to handle exception,we can handle exception by using try catch block

Eg:

```
public class ExceptionExample {
   public static void main(String[] args) {
        try{
            System.out.println(divide(4,0));
        }
        catch(ArithmeticException e){
            System.out.println("Can not divide by Zero :");
        }
   }
   public static int divide(int a, int b) throws ArithmeticException{
        int divesionResult=a/b;
        return divesionResult;
   }

}
```

2. What is an Exception? Explain about kinds of Exceptions

Exception is an unwanted or unExpected event occur during the execution of program i.e at run time it disrupts the normal flow of program

3. Differentiate between an Error and Exception

Error:A serious problem that a reasonable application should not try to catch

Exception : condition that a reasonable application should try to catch

4. Explain about Exception Hierarchy

All Exceptions and Errors are subclasses of Throwable class hierarchy which is base class of hierarchy, one branch is headed by Exception.This is class is used for exceptional conditions that user program should catch.Another branch error handled by JVM to indicate errors handled to do with run-time errors

5. Differentiate between throw and throws with examples

throw :

Throw keyword is used in method body to throw an exception

```
public class AgeException {
   public static void main(String[] args) {
        int age=18;
```

```
        try{
                if(age>=18){
                        throw new Except(18);
                }
        }
        catch(Exception e){
                System.out.println(e);
        }

   }

}

class Except extends Exception{
   int a;
        Except(int b) {
        a=b;
        }
        public String toString(){
        return ("Exception Number =  "+a) ;

        }
}
```

throws:

Throws keyword is used in method signature to declare the exception that occur in the statement present in the method

```
public class ThrowsExample {
   public static void main(String[] args) {
        try{
                System.out.println(divide(4,0));
        }
        catch(ArithmeticException e){
                System.out.println("Can not divide by Zero :");
        }
   }
   public static int divide(int a, int b) throws ArithmeticException{
        int divesionResult=a/b;
        return divesionResult;
```

```
    }

}
```

6. What are the different types of Exceptions in java? Explain in detail with examples
Types of Exceptions in java
1)checked Exceptions
2)un-checked Exceptions

1)Checked Exceptions:
Checked Exceptions are the exceptions that are handled at compile time, if some code with in the method throws checked exception then method must either handle the exception or must throw the exception using throws keyword
Eg:

```
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.io.Serializable;
import java.util.Scanner;

class Bank implements Serializable{
    private String accountNumber;
    private String accountName;
    private String userName;
    private String password;
    private float balance;
    public void setAccountNumber(String accountNumber){
        this.accountNumber=accountNumber;
    }
    public void setAccountName(String accountName){
        this.accountName=accountName;
    }
    public void setUserName(String userName){
        this.userName=userName;
    }
    public void setPassword(String password){
        this.password=password;
    }
    public void setBalance(float balance){
        this.balance=balance;
```

```
    }
    public String getAccountNumber(){
        return this.accountNumber;
    }
    public String getAccountName(){
        return this.accountName;
    }
    public String getPassword(){
        return this.password;
    }
    public String getUserName(){
        return this.userName;
    }
    public float getBalance(){
        return this.getBalance();
    }
}
public class BankTransactions {
   public static void main(String args[]) throws IOException{
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the number of bank employees :");
        int size=sc.nextInt();
        Bank customer[]=new Bank[size];
       FileOutputStream file = new FileOutputStream("sample.ser");
       ObjectOutputStream out = new ObjectOutputStream(file);
        for(int i=0;i<size;i++){
            customer[i]=new Bank();
            System.out.println("Enter the name of the account of employee no :"+(i+1));
            customer[i].setAccountName(sc.next());
            System.out.println("Enter the account number of bank employee :");
            customer[i].setAccountNumber(sc.next());
            System.out.println("Enter the user name of the Employee :");
            customer[i].setUserName(sc.next());
            System.out.println("Enter the password of the Employee :");
            customer[i].setPassword(sc.next());
            System.out.println("Enter the balance of the Employee :");
            customer[i].setBalance(sc.nextFloat());

        }
        out.writeObject(customer);
        out.close();
        file.close();
```

```
    }

}
```

In the above program we get checked exception while performing file operations so we throw the Exception by using throws keyword

2)un-checked Exceptions:
un-checked Exceptions are Exceptions that are handled at runtime,in java error and runtimeException class are un-checked Exception and every thing comes under checked Exception
Eg:

```
class Main {
   public static void main(String args[]) {
      int x = 0;
      int y = 10;
      int z = y/x;
  }
}
```

7. What is the output of the following programs ? Explain in detailed
(a)

```
import java.util.Scanner;

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

  int a, b, result;

  Scanner input = new Scanner(System.in);
  System.out.println("Input two integers");

  a = input.nextInt();
  b = input.nextInt();

  result = a / b;

  System.out.println("Result = " + result);
```

```
  }
}
```
Input 2,0
Output:program abnormally terminates as the Exception is occured while dividing by 0

(b)

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

  int a, b, result;

  Scanner input = new Scanner(System.in);
  System.out.println("Input two integers");

  a = input.nextInt();
  b = input.nextInt();

  try {
        result  = a / b;
        System.out.println("Result = " + result);
  }

  catch (ArithmeticException e) {
        System.out.println("Exception caught: Division by zero.");
  }
  }
}
```
Input : 2, 0
output: Exception caught: Division by zero.

(c)

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

  String languages[] = { "C", "C++", "Java", "Perl", "Python" };

  try {
        for (int c = 1; c <= 5; c++) {
        System.out.println(languages[c]);
```

```
        }
  }
  catch (Exception e) {
        System.out.println(e);
  }
  }
}
```

Output:

C++
Java
Perl
Python
java.lang.ArrayIndexOutOfBoundsException: 5

8)What is Serialization and DeSerialization in java? explain with examples

Serialization is a mechanism to convert object into stream of bytes, so that it can be written into files, can be transferred through network or stored into database. de serialization is vice versa which means converting stream of bytes into object.java serialization API provides features to perform serialization and deserialization. to perform serialization and deserialization a class must implement java.io.Serializable interface

9)What is Byte stream in java

Stream is a method for sequentially accessing files
Byte stream process the data byte by byte for example fileinputstream is used to read from source and fileoutputstream is used to write into destination

10. What is InputStream and What is OutputStream

Input stream : reads data from the source
Output stream : writes data into destination

11. Mention about the methods used in FileInputStream and FileOutputStream

Java FileInputStream obtains input from a file.FileInputStream is used to read stream of raw bytes such as image data , audio , video. We can read character data using FileInputStream, but for reading stream of characters FileReader is recommended

Methods of FileInputStream :

Int read():reads bytes of data from the input stream

Int read(byte b[]) : reads bytes of data from input stream up to b.length into array of bytes

int read(byte[] b, int off, int len) : Reads up to len bytes of data from this input stream into an array of bytes.

long skip(long n): skips over and discards n bytes of data from input stream

int available(): returns estimate of number of bytes that can be read from the input stream

void close(): closes the file input stream and releases any system resources associated with in the stream

FileDescriptor getFD() : returns the file description object that represents the connection to actual file in the file system being used by the FileInputStream

FileChannel getChannel() :returns the unique file channel object associated with the FileInputStream

void finalize() : Ensures that the close method of this file input stream is called when there are no more references to it.

Java FileOutputStream Reader used for writing stream of raw bytes such as images into file

Methods of FileOutputStream:

Void close():closes the file output stream and releases any system resources associated with in the stream

Protected void finalize(): cleans up the connections to the file and ensures that close method of FileOutputStream is called when there are no more references of the file

void write(byte[] b):reads b.length from specific byte array to the file output stream

void write(byte[] b, int off, int len) :writes len bytes from specified byte array starting at offset off to the OutputStream

void write(int b) :writes the specified byte to the fileOutputStream

12. Write a program for following
a. create a directory
b. In that directory add a file called text.txt

c. add content "This is a demo text file" in text.txt
d. Read the text.txt file

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Scanner;
public class Directory {
   public static void main(String[] args) throws IOException {
        Scanner sc=new Scanner(System.in);
        File file=new File("Directory");
        if(!file.isDirectory()){
                file.mkdir();
                System.out.println("Folder created succesfully :");
                System.out.println("No thing :");
        }

                file=new File("Directory/text.txt");
                OutputStream out = new FileOutputStream(file);
                System.out.println("Enter Input to write in a file : :");
                String str=sc.nextLine();
                for(int i=0;i<str.length();i++){
                        out.write(str.charAt(i));
                }

                out.close();
                InputStream in = new FileInputStream(file);
                int data=in.read();
                while(data!=-1){
                        System.out.print((char)data);
                        data=in.read();
                }
                in.close();

   }

}
```

13. Program to check if a file or directory physically exist or not.

```
import java.io.File;

public class FileIsExist {
   public static void main(String[] args) {
         File f=new File("Directory");
         if(f.exists()){
               System.out.println("Directory do exist :");
         }
         else{
               System.out.println("Directory is not present :");
         }
         f=new File("Directory/text.txt");
         if(f.exists()){
               System.out.println("File do exist :");
         }
         else{
               System.out.println("File is not present :");
         }
   }

}
```

14 Consider a file contains a paragraph of data. Write a program to read the file line by line

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Scanner;
public class Directory {
   public static void main(String[] args) throws IOException {
         Scanner sc=new Scanner(System.in);
         File file=new File("Directory/text.txt");
```

```
                InputStream in = new FileInputStream(file);
                int data=in.read();
                StringBuilder sb=new StringBuilder();
                while(data!=-1){
                        sb.append((char)data);
                        if((char)data=='\n'){
                                System.out.println(sb);
                                sb.delete(0, sb.length()-1);
                        }
                        data=in.read();
                }
                in.close();

    }

}
```

15. Mention about the methods in Java.io.File Class in Java

1. **boolean canExecute() :** Tests whether the application can execute the file denoted by this abstract pathname.
2. **boolean canRead()** : Tests whether the application can read the file denoted by this abstract pathname.
3. **boolean canWrite() :** Tests whether the application can modify the file denoted by this abstract pathname.
4. **int compareTo(File pathname) :** Compares two abstract pathnames lexicographically.
5. **boolean createNewFile() :** Atomically creates a new, empty file named by this abstract pathname .
6. **boolean mkdir() :** Creates the directory named by this abstract pathname.
7. **boolean delete() :** Deletes the file or directory denoted by this abstract pathname.
8. **boolean isFile() :** Tests whether the file denoted by this abstract pathname is a normal file.
9. **boolean isDirectory() :** Tests whether the file denoted by this pathname is a directory.

16.Create a class name Bank which contains the properties of account number, account name, user name, password, balance.
Create a class name BankTransaction where you create object for bank and add details of bank into the file(add atleast 3 bank objects).

```
import java.io.FileNotFoundException;
```

```
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.io.Serializable;
import java.util.Scanner;

class Bank implements Serializable{
    private String accountNumber;
    private String accountName;
    private String userName;
    private String password;
    private float balance;
    public void setAccountNumber(String accountNumber){
        this.accountNumber=accountNumber;
    }
    public void setAccountName(String accountName){
        this.accountName=accountName;
    }
    public void setUserName(String userName){
        this.userName=userName;
    }
    public void setPassword(String password){
        this.password=password;
    }
    public void setBalance(float balance){
        this.balance=balance;
    }
    public String getAccountNumber(){
        return this.accountNumber;
    }
    public String getAccountName(){
        return this.accountName;
    }
    public String getPassword(){
        return this.password;
    }
    public String getUserName(){
        return this.userName;
    }
    public float getBalance(){
        return this.balance;
    }
}
```

```
public class BankTransactions {
   public static void main(String args[]) throws IOException{
         Scanner sc=new Scanner(System.in);
         System.out.println("Enter the number of bank employees :");
         int size=sc.nextInt();
         Bank customer[]=new Bank[size];
        FileOutputStream file = new FileOutputStream("sample.ser");
        ObjectOutputStream out = new ObjectOutputStream(file);
         for(int i=0;i<size;i++){
                customer[i]=new Bank();
                System.out.println("Enter the name of the account of employee no :"+(i+1));
                customer[i].setAccountName(sc.next());
                System.out.println("Enter the account number of bank employee :");
                customer[i].setAccountNumber(sc.next());
                System.out.println("Enter the user name of the Employee :");
                customer[i].setUserName(sc.next());
                System.out.println("Enter the password of the Employee :");
                customer[i].setPassword(sc.next());
                System.out.println("Enter the balance of the Employee :");
                customer[i].setBalance(sc.nextFloat());

         }
         out.writeObject(customer);
         out.close();
         file.close();

   }

}
```

Create a class BankUser and do following operations

a. Read the bank details of user file.

b. Rename the username of the 2nd bank object

c. Remove the 3rd bank object from the file

```
import java.io.*;
import java.util.Scanner;
public class BankUser {
   public static void main(String args[]) throws IOException, ClassNotFoundException{
         Scanner sc=new Scanner(System.in);
         FileInputStream fin=new FileInputStream("sample.ser");
         ObjectInputStream oin=new ObjectInputStream(fin);
```

```
        Bank bank[]=(Bank[])oin.readObject();
        Bank updatedBank[]=new Bank[bank.length-1];
        for(Bank bank1 : bank){
                System.out.println(bank1.getAccountName());
        }
        System.out.println("Enter the object Number to delete :");
        int objectNumber=sc.nextInt();
        int count=-1;
        for(int i=0;i<bank.length;i++){
                if(i==1){
                        System.out.println("Enter the username to change for object no :"+i+"");
                        bank[i].setAccountName(sc.next());
                }
                if(i!=objectNumber){
                        updatedBank[++count]=bank[i];

                }
        }
        for(Bank bank1 : updatedBank){
                System.out.println(bank1);
        }
        FileOutputStream fout=new FileOutputStream("sample.ser");
        ObjectOutputStream out=new ObjectOutputStream(fout);
        out.writeObject(updatedBank);
        oin.close();
        fin.close();
        out.close();
        fout.close();
    }

}
```

17. What is garbage collection?

In java garbage collector frees up the heap memory by destroying the unreachable objects, unreachable objects are the ones that are no longer referenced by any part of the program

18. Write a program to perform garbage collection

```
public class Garbage1 {

   public static void main(String args []){
         String s=new String("abc");
         s=null;
         System.gc();
   }


}
```

19. Explain about the Ways for requesting JVM to run Garbage Collector with example

Ways for requesting JVM to run Garbage Collector

Once we made object eligible for garbage collector it may not destroy immediately by garbage collector, when ever JVM runs garbage collector program then only object will be destroyed

We can request JVM to run garbage collector by two ways

1)System.gc()
2)Runtime.getRuntime().gc()

1)**System.gc()** :System class contains static method gc for requesting JVM to run garbage collector
2)**Runtime.getRuntime().gc():**Runtime class allows application to interface with JVM in which application is running.hence by using its gc() method we can request JVM to call garbage collector

20. What is Enumeration? Explain with an example

Enum is a special data type that enables variables to be set of predefined constants.the variable must be equal to one of the variables that have been pre defined for it

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

21. Consider a variable Directions of enum type, which is a collection of four constants EAST, WEST, NORTH and SOUTH.
Create a class EnumDemo and make use of enum variable. (example if you get the enum value is EAST. you should display "you are at EAST direction")

```
public  class Enumerate
{
   String str;
   public enum Directions{

            EAST("you are at EAST direction"),
            WEST("you are at WEST direction"),
            NORTH("you are at NORTH direction"),
            SOUTH("you are at SOUTH direction");
            private final String str;
            Directions(String str){
                    this.str=str;
            }
            public void getDirections(){
                    System.out.println(str);
            }
   }

         public static void main(String[] args) {
          Directions d=Directions.EAST;
         d.getDirections();
         }
}
```

22. Explain about Autoboxing and Unboxing with an example

Autoboxing:converting primitive values into objects of the corresponding wrapper class is known as autoboxing

Unboxing:converting of objects of wrapper values into primitive type is called unboxing

Example:

```
public class AutoBoxingUnBoxing {
   public static void main(String[] args) {
          Integer i=10;
          /*
           * Below is auto boxing
```

```
         */
        int i1=Integer.valueOf(i);
        int i2=i.intValue();
        System.out.println(i1);
        System.out.println(i2);
        /*
         * Below is unboxing
         */
        Integer i3=Integer.valueOf(i1);
        System.out.println(i3);
    }

}
```

23. What are annotations in java? Explain with an example

Annotations are form of metadata provides information about program that are not part of program.they don't have effect on the operations of the code they annotate