/* File class and its methods */

```
/*
File f=new File("abc.txt");
• This line 1st checks whether abc.txt file is already available (or) not if it is already
available then "f" simply refers that file. If it is not already available then it won't create
any physical file just creates a java File object represents name of the file.
*/
/* 1. createNewFile() method */
/*
The createNewFile() method is used to create a new, empty file at the specified path.
It returns a boolean value indicating whether the file was successfully created.
Complete signature:
public boolean createNewFile() throws IOException
*/

import java.io.*;
class Program1
{
public static void main(String[] args) throws IOException
{
   File file= new File("cricket.txt");
   boolean fileCreated = file.createNewFile();
   if (fileCreated) {
      System.out.println("File created successfully.");
   } else {
      System.out.println("File already exists.");
   }
}
}
/* Output */
```

```
/*
 If the file is already present in Current Working Directory then the output is:
 File already exists.
 If the file is not present in Current Working Directory then the file is created the output is:
 File created successfully.
 */

 /*IMPORTANT NOTE:*/
 /*
  We can creae a file in specified path also.
  For that just replace file.txt by the path(say "E:\\anu.txt") in which you want to create a file.
  If I will take a path in which i want to create a file and path i have taken incorrect then we
  will get RunTime Exception.
  */

/* 2. exists() */
/*
 The exists() method in the File class in Java has the following signature:
 public boolean exists()
 This method does not take any arguments. It returns a boolean value indicating whether the file
 or directory represented by the File object exists in the file system. The return value is true
 if the file or directory exists, and false otherwise.
 */
import java.io.*;

class Program2 {
    public static void main(String[] args) throws IOException {
        File file = new File("file.txt");
       //file.createNewFile();
        boolean fileExists = file.exists();
        if (fileExists) {
```

```
            System.out.println("The file or directory exists.");
        } else {
            System.out.println("The file or directory does not exist.");
        }
    }
}
/* Output */
/*
 If file.txt is present in current working directory the output is:
 The file or directory exists.
 If file.txt is not present in current working directory the output is:
 The file or directory does not exist.
 */

 /* IMPORTANT NOTE: */
 /*
  If you want to check existence of file in the specified path then in "File file = new File("file.txt");",
replace
  file.txt by that path.
  */
/* 3. mkdir() method: */
/*
The mkdir() method is used to create a new directory at the specified path.
It returns a boolean value indicating whether the directory was successfully created.
Complete signature:
public boolean mkdir()
*/
import java.io.*;
class Program3
{
public static void main(String[] args) throws IOException
```

```
{
   /*
   File directory= new File("E:\\MyFolder1\\Myfoder2\\Myfolder3");
   File f1 = new File("E:\\MyFolder1","1.txt");
   f1.createNewFile();
   File f2 = new File("E:\\MyFolder1","2.txt");
   f2.createNewFile();
   File f3 = new File("E:\\MyFolder1\\MyFoder2","3.txt");
   f3.createNewFile();
   */
   File directory= new File("MyFolder1");
   boolean directoryCreated = directory.mkdir();
   if (directoryCreated) {
      System.out.println("Directory created successfully.");
   } else {
      System.out.println("Directory already exists.");
   }

}
}
/* Output */
/*
 If the path is not specified then the directory if it does not exist then created in current working directory.
 If the path is specified then the directory if it does not exist then created in the specified path.
 for the above program the output is:
 Directory already exists.
 */

/* File class constructors: */
/*
```

1) File f=new File(String name);

• Creates a java File object that represents name of the file or directory.

2) File f=new File(String subdirname,String name);

• Creates a File object that represents name of the file or directory present in specified sub directory.

3) File f=new File(File subdir,String name);

Creates a File object that represents name of the file or directory present in specified sub directory.

*/

/* Example of File Constructor1 */

/* Write code to create a file named with demo.txt in current working directory.*/

```
import java.io.*;
class Program4_FileConstructor1
{
public static void main(String[] args)throws IOException
{
File f=new File("demo.txt");
   boolean fileCreated = f.createNewFile();
   if (fileCreated) {
      System.out.println("File created successfully.");
   } else {
      System.out.println("File already exists.");
   }

}
}
```

/* Output */

/* File created successfully.*/

/* Example of File Constructor2 */

```
/* Write code to create a directory named with bhaskar123 in current working
directory and create a file named with abc.txt in that directory.*/
class Program4_FileConstructor2
{
public static void main(String[] args)throws IOException
{
File fl=new File("bhaskar123");
fl.mkdir();
File f2=new File("bhaskar123","abc.txt");
boolean fileCreatedNew =f2.createNewFile();
 if (fileCreatedNew) {
       System.out.println("File created successfully in specified folder");
    } else {
       System.out.println("File already exists in specified folder");
    }
}
}
 /* Output */
 /*
  File created successfully in specified folder
  */

  /* Example  of File Constructor3 */

/* Write code to create a directory named with Amit123 in D drive and create a file named with a.txt
in that directory.
use constructor 3 of File class*/
class Program4_FileConstructor3
{
public static void main(String[] args)throws IOException
```

```
{
File m=new File("D:\\Amit123");
m.mkdir();
File f2=new File(m,"a.txt");
boolean fileCreatedNew =f2.createNewFile();
 if (fileCreatedNew) {
       System.out.println("File created successfully in specified folder");
    } else {
       System.out.println("File already exists in specified folder");
    }
}
}
/* Output */
/*
 File created successfully in specified folder.
 Compile and run above program once again in that case output is:
 File already exists in specified folder.
 */

/* 4. isFile() method:*/
/*
The isFile() method is used to check if the File object represents a file or not.
if it is a file then it returns true otherwise false.
Syntax:
public boolean isFile()
*/

import java.io.*;

 class Program5 {
    public static void main(String[] args) throws IOException {
```

```
        File file = new File("D:\\java\\Java2_Programs\\Unit7_IO_Part1\\demo.txt");
        if (file.isFile()) {
            System.out.println("It is a file");
        } else {
            System.out.println("It is not a file");
        }
    }
}
/* Output */
/*
 It is a file
 */

/* 5. isDirectory() method:*/
/*
The isDirectory() method is used to check if the File object represents a directory or not.
if it is a directory then it returns true otherwise false.
Syntax:
public boolean isDirectory()
*/

import java.io.*;

 class Program6 {
    public static void main(String[] args) throws IOException {
        File file = new File("D:\\java\\Java2_Programs\\Unit7_IO_Part1\\demo.txt");
        if (file.isDirectory()) {
            System.out.println("It is a Directory");
        } else {
            System.out.println("It is not a Directory");
        }
```

```
    }
}
/* Output */
/*
 It is not a Directory
 */

/* 6. getPath() method: */
/*
The getPath() method in the File class of Java has the following signature:
public String getPath()
This method does not take any arguments and returns a String representing the pathname of the File object.
*/

import java.io.*;

 class Program7 {
    public static void main(String[] args) throws IOException {
        File file = new File("D:\\java\\Java2_Programs\\Unit7_IO_Part1\\monu.txt");
        file.createNewFile();
        String path=file.getPath();
        System.out.println("Path=" +path);
    }
}
/* Output */
/*
 Path=D:\java\Java2_Programs\Unit7_IO_Part1\monu.txt
 */

/* IMPORTANT NOTE:
```

The getPath() method returns the path exactly as it was specified when constructing the File object.

Therefore, if the File object was created with an absolute path, the getPath() method will return the absolute path as a String.

If the File object was created with a relative path, the getPath() method will return the relative path as a String.

It's important to note that the path returned by getPath() may not necessarily represent an existing file or directory in the file system.

It simply represents the path stored within the File object.

*/

/* 7. getAbsolutePath() method: */

/*

The getAbsolutePath() method in the File class of Java is used to retrieve

the absolute pathname represented by a File object.

It returns the absolute path in the form of a string, which represents the

complete and exact location of the file or directory in the file system.

Here's the complete signature of the getAbsolutePath() method:

public String getAbsolutePath()

This method does not take any arguments.

*/

```
import java.io.*;

class Program8 {
   public static void main(String[] args) throws IOException {
      File file1 = new File("Amit456");
      file1.mkdir();
      File file = new File("Amit456","nonu.txt");
      file.createNewFile();
      String path1=file.getPath();
      String path2=file.getAbsolutePath();
      System.out.println("Path1=" +path1);
```

```
        System.out.println("Path2=" +path2);
    }
}
/* Output */
/*
Path1=Amit456\nonu.txt
Path2=D:\java\Java2_Programs\Unit7_IO_Part1\Amit456\nonu.txt
 */

/* IMPORTANT NOTE:
 The getAbsolutePath() method returns the absolute pathname of the File object, which includes
 the complete and exact location of the file or directory in the file system. It resolves any
 relative paths and provides the full path starting from the root directory.
It's important to note that the path returned by getAbsolutePath() represents the actual location
of the file or directory in the file system, regardless of whether it exists or not.
 */


/* 8. getParent() method: */
/*
The getParent() method in the File class of Java is used to retrieve the parent directory
of a file or directory represented by a File object. It returns a String representing the
pathname of the parent directory.
Here's the complete signature of the getParent() method:
public String getParent()
This method does not take any arguments.
*/

import java.io.*;

 class Program9 {
```

```
    public static void main(String[] args) throws IOException {
        File parentDir1 = new File("Amit456");
        parentDir1.mkdir();
        File childFile1 = new File("Amit456","nonu.txt");
        childFile1.createNewFile();
        String path1=childFile1.getParent();
        System.out.println("Path1=" +path1);

/*
        File parentDr = new File("Arman");
        parentDr.mkdir();
        File childDr1 = new File(parentDr,"Lucky");
        childDr1.mkdir();
        File childDr2 = new File(childDr1,"Dhairya");
        childDr2.mkdir();
        File childFile = new File(childDr2,"son.txt");
        childFile.createNewFile();
        String path2=childFile.getParent();
        System.out.println("Path2=" +path2);

*/
        File f= new File("gungun.txt");
        f.createNewFile();
        String path3=f.getParent();
        System.out.println("Path3=" +path3);


        File f1= new File("bhaskar123\\abc.txt");
        String path4=f1.getParent();
        System.out.println("Path4=" +path4);
```

```
        File f2= new File("D:\\java\\Java2_Programs\\Unit7_IO_Part1\\bhaskar123\\abc.txt");
        String path5=f2.getParent();
        System.out.println("Path5=" +path5);


    }
}
/* Output */
/*
Path1=Amit456
Path2=Arman\Lucky\Dhairya
Path3=null
Path4=bhaskar123
Path5=D:\java\Java2_Programs\Unit7_IO_Part1\bhaskar123
 */

/* IMPORTANT NOTE:*/
/*
The getParent() method returns the pathname of the parent directory of the File object.
It retrieves the directory path that is one level above the file or directory represented by the File object.
It's important to note that if the File object represents a file or a directory without a parent,
or if the parent directory is not specified in the path, the getParent() method will return null.
 */



/* 9. getParentFile() method: */
/*
The getParentFile() method in the File class of Java is used to retrieve the parent directory
 of a file or directory represented by a File object. It returns a File object representing
 the parent directory.
```

Here's the complete signature of the getParentFile() method:

public File getParentFile()

This method does not take any arguments.

*/

```
import java.io.*;

 class Program10 {
    public static void main(String[] args) throws IOException {
        File parentDir1 = new File("Amit456");
        parentDir1.mkdir();
        File childFile1 = new File("Amit456","nonu.txt");
        childFile1.createNewFile();
        File path1=childFile1.getParentFile();
        System.out.println("Path1=" +path1);
/*

        File parentDr = new File("Arman");
        parentDr.mkdir();
        File childDr1 = new File(parentDr,"Lucky");
        childDr1.mkdir();
        File childDr2 = new File(childDr1,"Dhairya");
        childDr2.mkdir();
        File childFile = new File(childDr2,"son.txt");
        childFile.createNewFile();
        File path2=childFile.getParentFile();
        System.out.println("Path2=" +path2);
*/
        File f= new File("gungun.txt");
        f.createNewFile();
        File path3=f.getParentFile();
```

```
        System.out.println("Path3=" +path3);

        File f1= new File("bhaskar123\\abc.txt");
        File path4=f1.getParentFile();
        System.out.println("Path4=" +path4);

        File f2= new File("D:\\java\\Java2_Programs\\Unit7_IO_Part1\\bhaskar123\\abc.txt");
        File path5=f2.getParentFile();
        System.out.println("Path5=" +path5);

    }
}
/* Output */
/*
Path1=Amit456
Path2=Arman\Lucky\Dhairya
Path3=null
Path4=bhaskar123
Path5=D:\java\Java2_Programs\Unit7_IO_Part1\bhaskar123
 */

/* IMPORTANT NOTE:*/
/*
The getParentFile() method returns a File object representing the parent directory of the File object.
It retrieves the File object that corresponds to the directory that is one level above the file or
directory represented by the File object.
It's important to note that if the File object represents a file or a directory without a parent, or
if the parent directory is not specified in the path, the getParentFile() method will return null.
 */

/* 10 canExecute() method */
```

```
/*
The canExecute() method returns a boolean
value indicating whether the file or directory can be executed.
Here's the complete signature of the canExecute() method:
public boolean canExecute()
This method does not take any arguments.
*/

import java.io.*;
 class Program11 {
    public static void main(String[] args) throws IOException {

        File file1 = new File("D:\\z.txt");
        file1.createNewFile();

        if (file1.canExecute()) {
            System.out.println("The file or directory is executable.");
        } else {
            System.out.println("The file or directory is not executable.");
        }
    }
}
/* Output */
/*
 The file or directory is executable.
 */

/* 11. canRead() method */
/*
canRead() method is used to determine if a file or directory is readable.
Here's the complete signature of the canRead() method:
```

public boolean canRead()

This method does not take any arguments and returns a boolean value (true or false)

indicating whether the file or directory is readable.

*/

```
import java.io.*;


class Program12 {
    public static void main(String[] args)throws IOException {
        File file = new File("D:\\java\\Java2_Programs\\Unit7_IO_Part1\\gungun.txt");
        file.createNewFile();
        if (file.canRead()) {
            System.out.println("The file or directory is readable.");
        } else {
            System.out.println("The file or directory is not readable.");
        }
    }
}
```

/* Output */

/*

The file or directory is readable.

*/

/* 12. equals() method */

/*

The equals() method in the File class of Java is used to compare two File objects for equality.

It determines whether the File object on which the method is called and the specified object are equal.

Here's the complete signature of the equals() method:

public boolean equals(Object obj)

This method takes an Object as an argument and returns a boolean value indicating whether the two objects are equal.

*/

```
import java.io.*;


class Program13 {
   public static void main(String[] args) {
      File file1 = new File("D:\\java\\Java2_Programs\\Unit7_IO_Part1\\monu.txt");
      File file2 = new File("D:\\java\\Java2_Programs\\Unit7_IO_Part1\\monu.txt");
      File file3 = new File("D:\\java\\Java2_Programs\\Unit7_IO_Part1\\sunita.txt");
      boolean areEqual1 = file1.equals(file2);
      boolean areEqual2 = file1.equals(file3);
      System.out.println("Are file1 and file2 equal? " + areEqual1);
      System.out.println("Are file1 and file3 equal? " + areEqual2);
   }
}


/* Output */
/*
Are file1 and file2 equal? true
Are file1 and file3 equal? false
*/


/* 13.length() method */
/*
The length() method in the File class of Java is used to get the length (size) of a file in bytes.
 It returns a long value representing the length of the file.
Here's the complete signature of the length() method:
public long length()
This method does not take any arguments and returns a long value representing the length of the file.
*/
import java.io.*;;
class Program14 {
```

```
    public static void main(String[] args) {
        File file = new File("D:\\java\\Java2_Programs\\Unit7_IO_Part1\\sunita.txt"); //hello how r u? (content of file)
        long fileSize = file.length();
        System.out.println("File Size: " + fileSize + " bytes");
    }
}

/* Output */
/*
 File Size: 14 bytes
 */

/* 14. list() method */
/*
The list() method in the File class of Java is used to retrieve an array of strings
representing the names of files and directories in the specified directory.
It returns an array of type String containing the names of the files and directories
present in the directory.
Here's the complete signature of the list() method:
public String[] list()
This method does not take any arguments and returns an array of type String containing
the names of files and directories present in the specified directory.
*/
import java.io.*;
 class Program15 {
    public static void main(String[] args) throws IOException {
        File directory1 = new File("A");
        directory1.mkdir();
        File directory2 = new File("A","B");
        directory2.mkdir();
```

```
        File f1 = new File("A","1.txt");
        f1.createNewFile();
        File f2 = new File("A","2.txt");
        f2.createNewFile();
        String[] fileList = directory1.list();
        System.out.println("Files and Directories in " + directory1.getAbsolutePath() + ":");
        for (String file : fileList) {
           System.out.println(file);
        }
    }
}
/* Output */
/*
Files and Directories in D:\java\Java2_Programs\Unit7_IO_Part1\A:
1.txt
2.txt
B
 */

/* 15. listFiles() method */
/*
The listFiles() method in the File class of Java is used to retrieve an array of
File objects representing the files and directories in the specified directory.
It returns an array of type File containing the File objects representing the files
and directories present in the directory.
Here's the complete signature of the listFiles() method:
public File[] listFiles()
This method does not take any arguments and returns an array of type File containing
the File objects representing the files and directories present in the specified directory.
*/
import java.io.*;
```

```
 class Program16 {
    public static void main(String[] args) throws IOException {
        File directory1 = new File("A");
        directory1.mkdir();
        File directory2 = new File("A","B");
        directory2.mkdir();
        File f1 = new File("A","1.txt");
        f1.createNewFile();
        File f2 = new File("A","2.txt");
        f2.createNewFile();
        File[] fileList = directory1.listFiles();
        System.out.println("Files and Directories in " + directory1.getAbsolutePath() + ":");
        for (File file : fileList) {
            System.out.println(file);
        }
    }
}
/* Output */
/*
Files and Directories in D:\java\Java2_Programs\Unit7_IO_Part1\A:
A\1.txt
A\2.txt
A\B
 */

// Write a java program to count a  total number of files and directories in a given directory.
import java.io.*;
class Program17
{
    public static void main(String[] args) throws IOException{
        File f1=new File("JU");
```

```
        f1.mkdir();
        File f2=new File("JU","VU");
        f2.mkdir();
        File f3=new File("JU","SU");
        f3.mkdir();
        File f4=new File("JU","1.txt");
        f4.createNewFile();
        File f5=new File("JU","2.txt");
        f5.createNewFile();
        File f6=new File("JU","3.txt");
        f6.createNewFile();
        String[] files=f1.list();
        int count=0;
        for(String f:files)
        {
           System.out.println("The name of file or Directory is:"+ f);
           count++;
        }
        System.out.println("The Total number of file or Directory is: "+ count);

    }
}

/* Output */
/*
The name of file or Directory is:1.txt
The name of file or Directory is:2.txt
The name of file or Directory is:3.txt
The name of file or Directory is:SU
The name of file or Directory is:VU
The Total number of file or Directory is: 5
```

```
*/
// Write a java program to count a number of files and directories in a given directory.
//(Take two counts one for files and other for directories)
import java.io.*;
class Program18
{
   public static void main(String[] args) throws IOException{
      File f1=new File("JU");
      f1.mkdir();
      File f2=new File("JU","VU");
      f2.mkdir();
      File f3=new File("JU","SU");
      f3.mkdir();
      File f4=new File("JU","1.txt");
      f4.createNewFile();
      File f5=new File("JU","2.txt");
      f5.createNewFile();
      File f6=new File("JU","3.txt");
      f6.createNewFile();
      File[] files=f1.listFiles();
      int fileCount=0;
      int directoryCount=0;
      for(File f:files)
      {
         if(f.isFile())
         {
         System.out.println("The name of file is:"+ f);
         fileCount++;
         }
         else
         {
```

```
            System.out.println("The name of Directory is: "+ f);
            directoryCount++;
            }

        }
        System.out.println("Total number of files is:"+ fileCount);
        System.out.println("Total number of Directories is:"+ directoryCount);
}
}

/* Ouput */
/*
The name of file is:JU\1.txt
The name of file is:JU\2.txt
The name of file is:JU\3.txt
The name of Directory is: JU\SU
The name of Directory is: JU\VU
Total number of files is:3
Total number of Directories is:2
 */
```