by Kunal Sir

Assignment: File Handling

Part A: MCQ's

1. Which class is used to create a file in Java?

A. Scanner
B. File
C. FileInputStream
D. FileReader

Answer: B

---

2. Which method is used to create a new file?

A. create()
B. makeFile()
C. createNewFile()
D. newFile()

Answer: C

---

3. Which package contains file handling classes in Java?

A. java.util
B. java.lang
C. java.io
D. java.file

Answer: C

---

4. Which method checks whether a file exists?

A. isPresent()
B. available()
C. exists()
D. check()

Answer: C

---

5. Which class is used to write data into a file in byte format?

A. FileReader
B. FileWriter
C. FileOutputStream
D. Scanner

Answer: C

---

6. Which class is used to read data from a file in byte format?

A. FileReader
B. FileInputStream
C. Scanner
D. FileWriter

Answer: B

---

7. What does `file.length()` return?

A. Number of lines
B. File size in KB
C. File size in bytes
D. Number of characters

Answer: C

---

8. Which method checks whether a file can be read?

A. canWrite()
B. isRead()
C. canRead()
D. readable()

Answer: C

---

9. Which method checks whether a file can be written?

A. isWritable()
B. canWrite()
C. writeCheck()
D. writable()

Answer: B

---

10. Which method deletes a file in Java?

A. remove()
B. deleteFile()
C. delete()
D. erase()

Answer: C

---

11. What does `getName()` method return?

A. File path
B. Absolute path
C. File size
D. File name

Answer: D

---

12. Which method returns the absolute path of a file?

A. getPath()
B. getFilePath()
C. getAbsolutePath()
D. getLocation()

Answer: C

---

13. Which method checks whether a path is a directory?

A. isFile()
B. isDirectory()
C. isFolder()
D. checkDir()

Answer: B

---

14. Which class is used to read a file line by line (without buffered classes)?

A. BufferedReader
B. FileReader
C. Scanner
D. DataInputStream

Answer: C

---

15. Which constructor is used to append data to a file?

A. FileOutputStream("file.txt")
B. FileOutputStream("file.txt", true)
C. FileWriter("file.txt")
D. Scanner("file.txt")

Answer: B

---

16. What happens if `createNewFile()` is called on an existing file?

A. File is deleted
B. File is overwritten
C. Returns false
D. Throws error

Answer: C

---

17. Which method is used to check whether a path is a file?

A. isDirectory()
B. isFile()
C. isPath()
D. exists()

Answer: B

---

18. Which stream is used to copy data from one file to another?

A. FileReader & FileWriter
B. Scanner & PrintWriter
C. FileInputStream & FileOutputStream
D. DataInputStream & Scanner

Answer: C

---

19. What is the return type of `exists()` method?

A. int
B. String
C. boolean
D. void

Answer: C

---

20. Which exception is commonly handled in file handling programs?

A. ArithmeticException
B. IOException
C. NullPointerException
D. ArrayIndexOutOfBoundsException

Answer: B

Part B: Problem Statement

1. Create a File

Problem Statement:
Write a Java program to create a new file named `sample.txt`. If the file already exists, the program should display a message saying that the file already exists. If the file does not exist, it should be created successfully.

```
import java.io.*;
import java.util.Scanner;

class CreateFile {
    public static void main(String[] args) throws IOException {
        File file = new File("sample.txt");

        if (file.exists()) {
            System.out.println("File already exists");
        } else {
            file.createNewFile();
            System.out.println("File created successfully");
        }
    }
}
```

---

2. Check Whether File Exists

Problem Statement:
Write a Java program to check whether a file named `data.txt` exists in the given location. The program should print a message indicating whether the file is present or not.

```
import java.io.*;
import java.util.Scanner;

class CheckFileExists {
    public static void main(String[] args) {
        File file = new File("data.txt");

        if (file.exists()) {
            System.out.println("File is present");
        } else {
            System.out.println("File is not present");
        }
    }
}
```

3. Write Text into a File

Problem Statement:
Write a Java program to create a file named `hello.txt` and write the text "Hello Java" into that file. After writing, display a confirmation message.

```
import java.io.*;
import java.util.Scanner;

class WriteIntoFile {
    public static void main(String[] args) throws IOException {
        FileOutputStream fos = new FileOutputStream("hello.txt");
        String text = "Hello Java";

        fos.write(text.getBytes());
        fos.close();

        System.out.println("Text written successfully");
    }
}
```

---

4. Read Text from a File

Problem Statement:
Write a Java program to read the contents of a file named `hello.txt` and display the text on the console.

```
import java.io.*;
import java.util.Scanner;

class ReadFromFile {
    public static void main(String[] args) throws IOException {
        FileInputStream fis = new FileInputStream("hello.txt");
        int ch;

        while ((ch = fis.read()) != -1) {
            System.out.print((char) ch);
        }
        fis.close();
    }
}
```

5. Append Text to an Existing File

Problem Statement:
Write a Java program to add the text "Welcome to File Handling" at the end of an existing file without removing or overwriting the old content.

```
import java.io.*;
import java.util.Scanner;

class AppendToFile {
    public static void main(String[] args) throws IOException {
        FileOutputStream fos = new FileOutputStream("hello.txt",
true);
        String text = "\nWelcome to File Handling";

        fos.write(text.getBytes());
        fos.close();

        System.out.println("Text appended successfully");
    }
}
```

6. Read File Line by Line

Problem Statement:
Write a Java program to read a text file line by line and print each line separately on the console.

```
import java.io.*;
import java.util.Scanner;

class ReadLineByLine {
    public static void main(String[] args) throws Exception {
        File file = new File("hello.txt");
        Scanner sc = new Scanner(file);

        while (sc.hasNextLine()) {
            System.out.println(sc.nextLine());
        }
        sc.close();
    }
}
```

7. Count Number of Lines in a File

Problem Statement:
Write a Java program to count how many lines are present in a given text file and display the total count.

```
import java.io.*;
import java.util.Scanner;

class CountLinesInFile {
    public static void main(String[] args) throws Exception {
        File file = new File("hello.txt");
        Scanner sc = new Scanner(file);
        int count = 0;

        while (sc.hasNextLine()) {
            sc.nextLine();
            count++;
        }
        sc.close();

        System.out.println("Total number of lines: " + count);
    }
}
```

8. Check Read Permission of a File

Problem Statement:
Write a Java program to check whether a file has permission to be read. Display a message indicating whether the file is readable or not.

```
import java.io.*;
import java.util.Scanner;

class CheckReadPermission {
    public static void main(String[] args) {
        File file = new File("hello.txt");

        if (file.canRead()) {
            System.out.println("File is readable");
        } else {
            System.out.println("File is not readable");
        }
    }
}
```

9. Check Write Permission of a File

Problem Statement:
Write a Java program to check whether a file has permission to be written. Display a message indicating whether the file is writable or not.

```
import java.io.*;
import java.util.Scanner;

class CheckWritePermission {
    public static void main(String[] args) {
        File file = new File("hello.txt");

        if (file.canWrite()) {
            System.out.println("File is writable");
        } else {
            System.out.println("File is not writable");
        }
    }
}
```

10. Check Whether Path is File or Directory

Problem Statement:
Write a Java program to check whether a given path refers to a file or a directory. Display an appropriate message based on the result.

```
import java.io.*;
import java.util.Scanner;

class FileOrDirectoryCheck {
    public static void main(String[] args) {
        File file = new File("hello.txt");

        if (file.isFile()) {
            System.out.println("It is a file");
        } else if (file.isDirectory()) {
            System.out.println("It is a directory");
        } else {
            System.out.println("Path does not exist");
        }
    }
}
```


---

11. Display File Size

Problem Statement:
Write a Java program to find and display the size of a file in bytes.

```
import java.io.*;
import java.util.Scanner;

class DisplayFileSize {
    public static void main(String[] args) {
        File file = new File("hello.txt");
        System.out.println("File size: " + file.length() + "
bytes");
    }
}
```

---

12. Display File Name and Absolute Path

Problem Statement:
Write a Java program to display the name of a file and its absolute path on the system.

```
import java.io.*;
import java.util.Scanner;

class FileNameAndPath {
    public static void main(String[] args) {
        File file = new File("hello.txt");

        System.out.println("File Name: " + file.getName());
        System.out.println("Absolute Path: " +
file.getAbsolutePath());
    }
}
```

---

13. Read Only the First Line of a File

Problem Statement:
Write a Java program to read and display only the first line of a given text file.

```
import java.io.*;
import java.util.Scanner;

class ReadFirstLine {
    public static void main(String[] args) throws Exception {
        File file = new File("hello.txt");
        Scanner sc = new Scanner(file);

        if (sc.hasNextLine()) {
            System.out.println(sc.nextLine());
        }
        sc.close();
    }
}
```

14. Copy Content from One File to Another

Problem Statement:
Write a Java program to copy the content of one file into another file. The destination file should contain the same content as the source file.

```
import java.io.*;
import java.util.Scanner;

class CopyFileContent {
    public static void main(String[] args) throws IOException {
        FileInputStream fis = new FileInputStream("hello.txt");
        FileOutputStream fos = new FileOutputStream("copy.txt");

        int ch;
        while ((ch = fis.read()) != -1) {
            fos.write(ch);
        }

        fis.close();
        fos.close();

        System.out.println("File copied successfully");
    }
}
```

---

15. Delete a File

Problem Statement:
Write a Java program to delete a file named `old.txt`. Display a message showing whether the file was deleted successfully or not.

```
import java.io.*;
import java.util.Scanner;

class DeleteFile {
    public static void main(String[] args) {
        File file = new File("old.txt");

        if (file.delete()) {
            System.out.println("File deleted successfully");
        } else {
            System.out.println("File deletion failed");
        }
    }
}
```