Class Work - 4

Q. ExcDemo1

```java
class ExcDemo1 {
    public static void main(String[] args) {
        int[] numbers = new int[9];
        try {
            System.out.println("Before Exception");
            numbers[10] = 200;
            System.out.println("this won't be displayed");
        }
        catch(ArrayIndexOutOfBoundsException exc) {
            System.out.println("index out-of bound !");
        }
        System.out.println("After catch statement");
    }
}
```

Output :

PS D:\MCA\MCA SEM 3\JAVA\Practice> java .\ExcDemo1.java

Before Exception

index out-of bound !

After catch statement

Q. ExcDemo2

```java
public class ExcDemo2 {
    public static void main(String[] args) {
        int[] numberSet1 = { 12, 14, 12, 6, 18, 40 };
        int[] numberSet2 = { 2, 4, 0, 6, 0, 4 };
        try {
            System.out.println("Before Exception");
            for (int i = 0; i < numberSet1.length; i++) {
                System.out.println(numberSet1[i] / numberSet2[i]);
            }
            System.out.println("this won't be displayed");
        }
        catch(ArithmeticException exc) {
            System.out.println("Divide by Zero !");
        }
        System.out.println("After catch statement");
    }
}
```

Output :

PS D:\MCA\MCA SEM 3\JAVA\Practice> java .\ExcDemo2.java

Before Exception

6

3

Divide by Zero !

After catch statement

Q. Finally Demo

```
public class FinallyDemo {
    public static void main(String[] args) {
        for (int i = 0; i < 3; i++) {
            UseFinally.genException(i);
            System.out.println();
        }
    }
}

class UseFinally {
    public static void genException(int what) {
        int t;
        int nums[] = new int[2];
        System.out.println("Receiveing " + what);
        try {
            switch (what) {
                case 0:
                    t = 10 / what;
                    break;
                case 1:
                    nums[4] = 4;
                    break;
                case 2:
                    return;
            }
        } catch (ArithmeticException e) {
            System.out.println(e.getMessage());
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println(e.getMessage());
        } finally {
            System.out.println("Leaving Try");
        }
    }
}
```

Output :

PS D:\MCA\MCA SEM 3\JAVA\Practice> java .\FinallyDemo.java

Receiveing 0

/ by zero

Leaving Try

Receiveing 1

Index 4 out of bounds for length 2

Leaving Try

Receiveing 2

Leaving Try

Q. ThrowsDemo

```java
import java.io.IOException;

public class ThrowsDemo {
    public static char prompt(String str) throws IOException {
        System.out.print(str + " : ");
        return (char) System.in.read();
    }

    public static void main(String[] args) {
        char ch;

        try {
            ch = prompt("Enter a letter");
        } catch (IOException e) {
            System.out.println("I/O Exception occoured");
            ch = 'X';
        }
        System.out.println("You pressed " + ch);
    }
}
```

Output:

PS D:\MCA\MCA SEM 3\JAVA\Practice> java .\ThrowsDemo.java

Enter a letter : R

You pressed R

Q. MultiCatch

```java
public class MultiCatch {
    public static void main(String[] args) {
        int a = 88, b = 0;
        int result;
        char ch[] = { 'A', 'B', 'C' };

        for (int i = 0; i < 2; i++) {
            try {
                if (i == 0)
                    result = a / b;
                else
                    ch[5] = 'X';
            } catch (ArithmeticException | ArrayIndexOutOfBoundsException e) {
                System.out.println(e.getMessage());
            }
        }
        System.out.println("After multi Catch");
    }
}
```

Output:

PS D:\MCA\MCA SEM 3\JAVA\Practice> java .\MultiCatch.java

/ by zero

Index 5 out of bounds for length 3

After multi Catch

Q. Custom ExceptionsDemo

```
public class CustomExceptionDemo {
    public static void main(String[] args) {
        int numer[] = { 4, 8, 15, 32, 64, 128, 256, 512 };
        int denom[] = { 2, 0, 4, 4, 0, 8 };

        for (int i = 0; i < numer.length; i++) {
            try {
                if (numer[i] % 2 != 0)
                    throw new NonIntResultException(numer[i], denom[i]);

                System.out.println(numer[i] + " / " + denom[i] + " is " + nume
r[i] / denom[i]);
            } catch (ArithmeticException e) {
                System.out.println("Cannot divide by Zero");
            } catch (ArrayIndexOutOfBoundsException e) {
                System.out.println("No matching element found");
            } catch (NonIntResultException e) {
                System.out.println(e);
            }
        }
    }
}
class NonIntResultException extends Exception {
    int n, d;

    NonIntResultException(int i, int j) {
        n = i;
        d = j;
    }

    public String toString() {
        return "Result of " + n + " / " + d + " is non integer";
    }
}
```

Output :

PS D:\MCA\MCA SEM 3\JAVA\Practice> java .\CustomExceptionDemo.java

4 / 2 is 2

Cannot divide by Zero

Result of 15 / 4 is non integer

32 / 4 is 8

Q. Add exception to queue

```
class QueueFullException extends Exception {
    int size;

    QueueFullException(int s) {
        size = s;
    }

    public String toString() {
        return "\nQueue is full. Maximum size is " + size;
    }
}

class QueueEmptyException extends Exception {
    QueueEmptyException() {
    }

    public String toString() {
        return "\nQueue is Empty.";
    }
}

class Queue {
    private char[] q;
    private int putloc, getloc;

    Queue(int size) {
        q = new char[size];
        putloc = getloc = 0;
    }

    void put(char ch) throws QueueFullException {
        if (putloc == q.length) {
            throw new QueueFullException(q.length);
        }
        q[putloc++] = ch;
    }

    char get() throws QueueEmptyException {
        if (getloc == putloc)
            throw new QueueEmptyException();

        return q[getloc++];
    }
}

public class QueueDemo {
```

```
    public static void main(String[] args) throws Exception {
        Queue bigQ = new Queue(100);
        Queue smallQ = new Queue(4);
        char ch;
        int i;

        for (i = 0; i < 26; i++)
            bigQ.put((char) ('A' + i));

        for (i = 0; i < 26; i++) {
            ch = bigQ.get();
            if (ch != (char) 0)
                System.out.print(ch);
        }

        System.out.println("\n");

        System.out.println("Using smallQ to generate errors.");
        for (i = 0; i < 5; i++) {
            System.out.println("Attempting to store " + (char) ('Z' - i));

            smallQ.put((char) ('Z' - i));

            System.out.println();
        }
        System.out.println();

        System.out.println("Contents of SmallQ: ");
        for (i = 0; i < 5; i++) {
            ch = smallQ.get();
            if (ch != (char) 0)
                System.out.print(ch);
        }
    }
}
```

Output :

PS D:\MCA\MCA SEM 3\JAVA\Practice> java .\QueueDemo.java

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Using smallQ to generate errors.

Attempting to store Z

Attempting to store Y

Attempting to store X

Attempting to store W

Attempting to store V

Exception in thread "main"

Queue is full. Maximum size is 4

at Queue.put(QueueDemo.java:33)

at QueueDemo.main(QueueDemo.java:68)

Q. Try It Out

```
class ZeroDivideException extends Exception {
    private int index = -1;

    public ZeroDivideException() {
    }

    public ZeroDivideException(String s) {
        super(s);
    }

    public ZeroDivideException(int index) {
        super("/ by zero");
        this.index = index;
    }

    public int getIndex() {
        return index;
    }
}

public class TryItOut {
    public static int divide(int[] array, int index) throws ZeroDivideExceptio
n {
        try {
            System.out.println("First try block in divide() entered");
            array[index + 2] = array[index] / array[index + 1];
            System.out.println("Code at end of first try block in divide()");
            return array[index + 2];
        } catch (ArithmeticException e) {
            System.out.println("Arithmetic exception caught in divide()");
            throw new ZeroDivideException(index + 1);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Index-out-of-
bounds index exception caught in divide()");
        }
        System.out.println("Executing code after try block in divide()");
        return array[index + 2];
    }

    public static void main(String[] args) {
        int[] x = { 10, 5, 0 };
        try {
            System.out.println("First try block in main()entered");
            System.out.println("result = " + divide(x, 0));
            x[1] = 0;
            System.out.println("result = " + divide(x, 0));
```

```
            x[1] = 1;
            System.out.println("result = " + divide(x, 1));
        } catch (ZeroDivideException e) {
            int index = e.getIndex();
            if (index > 0) {
                x[index] = 1;
                x[index + 1] = x[index - 1];
                System.out.println("Zero divisor corrected to " + x[index]);
            }
        } catch (ArithmeticException e) {
            System.out.println("Arithmetic exception caught in main()");
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Index-out-of-
bounds exception caught in main()");
        }
        System.out.println("Outside first try block in main()");
    }
}
```

Output :

PS D:\MCA\MCA SEM 3\JAVA\Practice> java .\TryItOut.java

First try block in main()entered

First try block in divide() entered

Code at end of first try block in divide()

result = 2

First try block in divide() entered

Arithmetic exception caught in divide()

Zero divisor corrected to 1

Outside first try block in main()

Q. Java 368

```java
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;

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

        int i;
        FileInputStream fin;

        if (args.length != 1) {
            System.out.println("Usgae: ShowFile File");
            return;
        }

        try {
            fin = new FileInputStream(args[0]);
        } catch (FileNotFoundException e) {
            System.out.println(e.getMessage());
            return;
        }

        try {
            do {
                i = fin.read();
                if (i != -1)
                    System.out.print((char) i);
            } while (i != -1);
        } catch (IOException e) {
            System.out.println(e.getMessage());
        }

        try {
            fin.close();
        } catch (IOException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Output :

PS D:\MCA\MCA SEM 3\JAVA\Practice> java .\ShowFile.java new.txt

Some data in file

Q. Help -> 10.2

```
import java.io.*;

class Help {
    String helpfile;

    Help(String fname) {
        helpfile = fname;
    }

    boolean helpOn(String what) {
        int ch;
        String topic, info;

        try (BufferedReader helprdr = new BufferedReader(new FileReader(helpfi
le))) {
            do {
                ch = helprdr.read();
                if (ch == '#') {
                    topic = helprdr.readLine();
                    if (what.compareTo(topic) == 0) {
                        do {
                            info = helprdr.readLine();
                            if (info != null)
                                System.out.println(info);
                        } while ((info != null) && (info.compareTo("") != 0));
                        return true;
                    }
                }
            } while (ch != 1);
        } catch (IOException e) {
            System.out.println(e.getMessage());
            return false;
        }
        return false;
    }

    String getSelection() {
        String topic = "";
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in
));
        System.out.print("Enter topic: ");
        try {
            topic = br.readLine();
        } catch (IOException e) {
            System.out.println(e.getMessage());
        }
```

```
        return topic;
    }
}

public class FileHelp {
    public static void main(String[] args) {
        Help hlpobj = new Help("helpfile.txt");
        String topic;

        System.out.println("try the help system." + "Enter 'stop' to end");

        do {
            topic = hlpobj.getSelection();

            if (!hlpobj.helpOn(topic))
                System.out.println("Topic not found");

        } while (topic.compareTo("stop") != 0);
    }
}
```

Output :

PS D:\MCA\MCA SEM 3\JAVA\Practice> java .\FileHelp.java

try the help system.Enter 'stop' to end

Enter topic: topic2

this is help on topic 2

Enter topic: topic1

this is help on topic 1

Q. Write A String

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.FileNotFoundException;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;

public class WriteAString {
    public static void main(String[] args) {
        String phrase = new String("Garbage in, garbage out\n");
        String dirname = "D:/New";
        String filename = "charData.txt";
        File dir = new File(dirname);

        if (!dir.exists()) {
            if (!dir.mkdir()) {
                System.out.println("Cannot create directory: " + dirname);
                System.exit(1);
            }
        } else if (!dir.isDirectory()) {
            System.err.println(dirname + " is not a directory");
            System.exit(1);
        }
        File aFile = new File(dir, filename);
        FileOutputStream outputFile = null;
        try {
            outputFile = new FileOutputStream(aFile, true);
            System.out.println("File stream created successfully.");
        } catch (FileNotFoundException e) {
            e.printStackTrace(System.err);
        }

        FileChannel outChannel = outputFile.getChannel();
        ByteBuffer buf = ByteBuffer.allocate(1024);
        System.out.println("New buffer: position = " + buf.position() + "\tLim
it = " + buf.limit() + "\tcapacity = "
                + buf.capacity());

        for (char ch : phrase.toCharArray()) {
            buf.putChar(ch);
        }
        System.out.println("Buffer after loading: position = " + buf.position(
) + "\tLimit = " + buf.limit()
                + "\tcapacity = " + buf.capacity());
        buf.flip();
```

```
        System.out.println("Buffer after flip: position = " + buf.position() +
 "\tLimit = " + buf.limit()
                + "\tcapacity = " + buf.capacity());

        try {
            outChannel.write(buf);
            outputFile.close();
            System.out.println("Buffer contents written to file.");
        } catch (IOException e) {
            e.printStackTrace(System.err);
        }
        System.exit(0);
    }
}
```

Output :

PS D:\MCA\MCA SEM 3\JAVA\Practice> java .\WriteAString.java

File stream created successfully.

New buffer: position = 0 Limit = 1024 capacity = 1024

Buffer after loading: position = 48 Limit = 1024 capacity = 1024

Buffer after flip: position = 0 Limit = 48 capacity = 1024

Buffer contents written to file.

Q. Write A String As Byte

```java
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.FileNotFoundException;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;

public class WriteAStringAsBytes {
    public static void main(String[] args) {
        String phrase = new String("Garbage in, garbage out\n");
        String dirname = "D:/New";
        String filename = "byteData.txt";
        File aFile = new File(dirname, filename);

        FileOutputStream file = null;
        try {
            file = new FileOutputStream(aFile, true);
        } catch (FileNotFoundException e) {
            e.printStackTrace(System.err);
        }
        FileChannel outChannel = file.getChannel();
        ByteBuffer buf = ByteBuffer.allocate(phrase.length());
        byte[] bytes = phrase.getBytes();
        buf.put(bytes);
        buf.flip();
        try {
            outChannel.write(buf);
            file.close();
        } catch (IOException e) {
            e.printStackTrace(System.err);
        }
    }
}
```

Run java file :

PS D:\MCA\MCA SEM 3\JAVA\Practice> java .\WriteAStringAsByte.java

Q. Proverbs

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.FileNotFoundException;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;

public class WriteProverbs {
    public static void main(String[] args) {
        String dirName = "D:/New";
        String fileName = "Proverbs.txt";
        String[] sayings = { "Indecision maximizes flexibility.", "Only the me
diocre are always at their best.",
                "A little knowledge is a dangerous thing.", "Many a mickle mak
es a muckle.",
                "Who begins too much achieves little.", "Who knows most says l
east.",
                "A wise man sits on the hole in his carpet." };
        File aFile = new File(dirName, fileName);
        FileOutputStream outputFile = null;
        try {
            outputFile = new FileOutputStream(aFile, true);
        } catch (FileNotFoundException e) {

            e.printStackTrace(System.err);
            System.exit(1);
        }
        FileChannel outChannel = outputFile.getChannel();

        int maxLength = 0;
        for (String saying : sayings) {
            if (maxLength < saying.length())
                maxLength = saying.length();
        }
        ByteBuffer buf = ByteBuffer.allocate(2 * maxLength + 4);

        try {
            for (String saying : sayings) {
                buf.putInt(saying.length()).asCharBuffer().put(saying);
                buf.position(buf.position() + 2 * saying.length()).flip();
                outChannel.write(buf);
                buf.clear();
            }
            outputFile.close();
            System.out.println("Proverbs written to file.");
        } catch (IOException e) {
```

```
            e.printStackTrace(System.err);
            System.exit(1);
        }
        System.exit(0);
    }
}
```

Output :

PS D:\MCA\MCA SEM 3\JAVA\Practice> java .\WriteProverbs.java

Proverbs written to file

Q. Use A Formatter

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.FileNotFoundException;
import java.nio.ByteBuffer;
import java.nio.CharBuffer;
import java.nio.channels.FileChannel;
import java.util.Formatter;

public class UsingAFormatter {
    public static void main(String[] args) {
        String[] phrases = { "Rome wasn't burned in a day.", "It's a bold mous
e that sits in the cat's ear.",
                "An ounce of practice is worth a pound of instruction." };
        String dirname = "D:/New";
        String filename = "Phrases.txt";
        File dir = new File(dirname);

        if (!dir.exists()) {
            if (!dir.mkdir()) {
                System.out.println("Cannot create directory: " + dirname);
                System.exit(1);
            }
        } else if (!dir.isDirectory()) {
            System.err.println(dirname + " is not a directory");
            System.exit(1);
        }

        File aFile = new File(dir, filename);
        FileOutputStream outputFile = null;
        try {
            outputFile = new FileOutputStream(aFile, true);
            System.out.println("File stream created successfully.");
        } catch (FileNotFoundException e) {
            e.printStackTrace(System.err);
        }

        FileChannel outChannel = outputFile.getChannel();

        ByteBuffer buf = ByteBuffer.allocate(1024);
        System.out.println("\nByte buffer:");
        System.out.printf("position = %2d Limit = %4d capacity = %4d%n", buf.p
osition(), buf.limit(), buf.capacity());

        CharBuffer charBuf = buf.asCharBuffer();
```

```
        System.out.println("Char view buffer:");
        System.out.printf("position = %2d Limit = %4d capacity = %4d%n", charB
uf.position(), charBuf.limit(),
                charBuf.capacity());
        Formatter formatter = new Formatter(charBuf);

        int number = 0;
        for (String phrase : phrases) {
            formatter.format("%nProverb%3d: %s", ++number, phrase);
            System.out.println("\nView buffer after loading:");
            System.out.printf("position = %2d Limit = %4d capacity = %4d%n", c
harBuf.position(), charBuf.limit(),
                    charBuf.capacity());
            charBuf.flip();
            System.out.println("View buffer after flip:");
            System.out.printf("position = %2d Limit = %4d length = %4d%n", cha
rBuf.position(), charBuf.limit(),
                    charBuf.length());
            buf.limit(2 * charBuf.length());
            System.out.println("Byte buffer after limit update:");
            System.out.printf("position = %2d Limit = %4d length = %4d%n", buf
.position(), buf.limit(),
                    buf.remaining());

            try {
                outChannel.write(buf);
                System.out.println("Buffer contents written to file.");
                buf.clear();
                charBuf.clear();
            } catch (IOException e) {
                e.printStackTrace(System.err);
                System.exit(1);
            }
        }
        try {
            outputFile.close();
        } catch (IOException e) {
            e.printStackTrace(System.err);
        }
    }
}
```

Output :

PS D:\MCA\MCA SEM 3\JAVA\Practice> java .\UsingAFormatter.java
File stream created successfully.

Byte buffer:
position = 0 Limit = 1024 capacity = 1024
Char view buffer:
position = 0 Limit = 512 capacity = 512

View buffer after loading:
position = 44 Limit = 512 capacity = 512
View buffer after flip:
position = 0 Limit = 44 length = 44
Byte buffer after limit update:
position = 0 Limit = 88 length = 88
Buffer contents written to file.

View buffer after loading:
position = 63 Limit = 512 capacity = 512
View buffer after flip:
position = 0 Limit = 63 length = 63
Byte buffer after limit update:
position = 0 Limit = 126 length = 126
Buffer contents written to file.

View buffer after loading:
position = 67 Limit = 512 capacity = 512
View buffer after flip:
position = 0 Limit = 67 length = 67
Byte buffer after limit update:
position = 0 Limit = 134 length = 134
Buffer contents written to file.

Q. Try It Out Copying File

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.nio.channels.FileChannel;

public class FileCopy {
    public static void main(String[] args) {
        if (args.length == 0) {
            System.out.println("No file to copy. Application usage is:\n" + "j
ava -classpath . FileCopy \"filepath\"");
            System.exit(1);
        }
        File fromFile = new File(args[0]);
        if (!fromFile.exists()) {
            System.out.printf("File to copy, %s, does not exist.", fromFile.ge
tAbsolutePath());
            System.exit(1);
        }
        File toFile = createBackupFile(fromFile);
        FileInputStream inFile = null;
        FileOutputStream outFile = null;
        try {
            inFile = new FileInputStream(fromFile);
            outFile = new FileOutputStream(toFile);
        } catch (FileNotFoundException e) {
            e.printStackTrace(System.err);
            assert false;
        }
        FileChannel inChannel = inFile.getChannel();
        FileChannel outChannel = outFile.getChannel();
        try {
            int bytesWritten = 0;
            long byteCount = inChannel.size();
            while (bytesWritten < byteCount) {
                bytesWritten += inChannel.transferTo(bytesWritten, byteCount -
 bytesWritten, outChannel);
            }
            System.out.printf("File copy complete. %d bytes copied to %s%n", b
yteCount, toFile.getAbsolutePath());
            inFile.close();
            outFile.close();
        } catch (IOException e) {
            e.printStackTrace(System.err);
            System.exit(1);
```

```
        }
        System.exit(0);
    }

    public static File createBackupFile(File aFile) {
        aFile = aFile.getAbsoluteFile();
        File parentDir = new File(aFile.getParent());
        String name = aFile.getName();
        int period = name.indexOf('.');
        if (period == -1) {
            period = name.length();
        }
        String nameAdd = "_backup";
        File backup = aFile;
        while (backup.exists()) {
            name = backup.getName();
            backup = new File(parentDir, name.substring(0, period) + nameAdd +
 name.substring(period));
            period += nameAdd.length();
        }
        return backup;
    }
}
```

Output :

PS D:\MCA\MCA SEM 3\JAVA\Practice> java .\FileCopy.java byteData.txt
File copy complete. 12 bytes copied to D:\MCA\MCA SEM 3\JAVA\Practice\byteData_backup.txt

Q. Read & Write

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;

public class RandomReadWrite {
    public static void main(String[] args) {
        File aFile = new File("primes.txt");
        FileInputStream inFile = null;
        FileOutputStream outFile = null;
        try {
            inFile = new FileInputStream(aFile);
            outFile = new FileOutputStream(aFile, true);
        } catch (FileNotFoundException e) {
            e.printStackTrace(System.err);
            System.exit(1);
        }
        FileChannel inChannel = inFile.getChannel();
        FileChannel outChannel = outFile.getChannel();
        final int PRIMESREQUIRED = 10;
        ByteBuffer buf = ByteBuffer.allocate(8);
        long[] primes = new long[PRIMESREQUIRED];
        int index = 0;
        final long REPLACEMENT = 99999L;
        try {
            final int PRIMECOUNT = (int) inChannel.size() / 8;
            System.out.println("Prime count = " + PRIMECOUNT);
            for (int i = 0; i < PRIMESREQUIRED; i++) {
                index = 8 * (int) (PRIMECOUNT * Math.random());
                inChannel.read(buf, index);
                buf.flip();
                primes[i] = buf.getLong();
                buf.flip();
                buf.putLong(REPLACEMENT);
                buf.flip();
                outChannel.write(buf, index);
                buf.clear();
            }
            int count = 0;
            for (long prime : primes) {
                System.out.printf("%12d", prime);
                if (++count % 5 == 0) {
                    System.out.println();
```

```
                }
            }
            inFile.close();
            outFile.close();
        } catch (IOException e) {
            e.printStackTrace(System.err);
            System.exit(1);
        }
        System.exit(0);
    }
}
```

Output :

359 107 383 109 7
173 443 337 17 113

Q. Use Threads

```
class UseThreads {
    public static void main(String[] args) {
        System.out.println("Main thread starting.");

        MyThread mt = new MyThread("Child #1");

        Thread newThrd = new Thread(mt);

        newThrd.start();

        for(int i=0;i<50;i++)
        {
            System.out.println(".\n" + newThrd.isAlive());
            System.out.println(".");
            try{
                Thread.sleep(200);
            }
            catch(InterruptedException exc){
                System.out.println("Main thread interrupted.");
            }
            newThrd.interrupt();
        }
        System.out.println("Main thread ending.");
    }
}

class MyThread implements Runnable{
    String thrdName;
    MyThread(String name) {
        thrdName = name;
    }

    public void run() {
        System.out.println(thrdName + "starting.");
        try {
            for(int count=0;count<10;count++) {
                Thread.sleep(5000);
                System.out.println("In " + thrdName + ", count is " +count);
            }
        }
        catch(InterruptedException exc) {
            System.out.println(thrdName + " interrupted.");
        }

        System.out.println(thrdName + " terminating.");
    }
```

```
}
```

Output :

PS D:\MCA\MCA SEM 3\JAVA\Thread> java .\UseThreads.java
Main thread starting.
.
true
.
Child #1starting.
.
true
.
Child #1 interrupted.
Child #1 terminating.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false

.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.

.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.

false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
.
false
.
Main thread ending.

Q. Sum Array Synchronization

```
class SumArray {
    private int sum;

    synchronized int sumArray(int[] nums) {
        sum = 0;

        for (int i = 0; i < nums.length; i++) {
            sum += nums[i];
            System.out.println("Running total for " + Thread.currentThread().g
etName() + " is " + sum);

            try {
                Thread.sleep(10);
            } catch (InterruptedException exc) {
                System.out.println("Thread interrupted!");
            }
        }
        return sum;
    }
}

class MyThread implements Runnable {
    Thread thrd;
    static SumArray sa = new SumArray();
    int[] a;
    int answer;

    MyThread(String name, int[] nums) {
        thrd = new Thread(this, name);
        a = nums;
        thrd.start();
    }

    public void run() {
        int sum;

        System.out.println(thrd.getName() + " starting...");

        answer = sa.sumArray(a);
        System.out.println("Sum for " + thrd.getName() + " is " + answer);

        System.out.println(thrd.getName() + " terminating...");
    }
}

public class SumArraySync {
```

```
    public static void main(String[] args) {
        int[] a = { 1, 2, 3, 4, 5 };

        MyThread mt1 = new MyThread("Child #1", a);
        MyThread mt2 = new MyThread("Child #2", a);

        try {
            mt1.thrd.join();
            mt2.thrd.join();
        } catch (InterruptedException e) {
            System.out.println("Main thread interrupted!");
        }
    }
}
```

Output :

PS D:\MCA\MCA SEM 3\JAVA> java .\SumArraySync.java
Child #2 starting...
Child #1 starting...
Running total for Child #2 is 1
Running total for Child #2 is 3
Running total for Child #2 is 6
Running total for Child #2 is 10
Running total for Child #2 is 15
Running total for Child #1 is 1
Sum for Child #2 is 15
Child #2 terminating...
Running total for Child #1 is 3
Running total for Child #1 is 6
Running total for Child #1 is 10
Running total for Child #1 is 15
Sum for Child #1 is 15
Child #1 terminating...

Q. Class TikTok

```
public class ThreadCom {
    public static void main(String[] args) {
        TickTock t1 = new TickTock();
        MyThread mt1 = new MyThread("Tick", t1);
        MyThread mt2 = new MyThread("Tock", t1);

        try {
            mt1.thrd.join();
            mt2.thrd.join();
        } catch (InterruptedException e) {
            System.out.println("Main thread interrupted...");
        }
    }
}

class TickTock {
    String state;

    synchronized void tick(boolean running) {
        if (!running) {
            state = "ticked";
            notify();
            return;
        }

        System.out.println("Tick ");

        state = "ticked";
        notify();

        try {
            while (!state.equals("tocked"))
                wait();
        } catch (InterruptedException e) {
            System.out.println("Thread interrupted");
        }
    }

    synchronized void tock(boolean running) {
        if (!running) {
            state = "tocked";
            notify();
            return;
        }

        System.out.println("Tock ");
```

```
        state = "tocked";
        notify();

        try {
            while (!state.equals("ticked"))
                wait();
        } catch (InterruptedException e) {
            System.out.println("Thread interrupted");
        }
    }
}

class MyThread implements Runnable {
    Thread thrd;
    TickTock ttOb;

    MyThread(String name, TickTock tt) {
        thrd = new Thread(this, name);
        ttOb = tt;
        thrd.start();
    }

    public void run() {
        if (thrd.getName().compareTo("Tick") == 0) {
            for (int i = 0; i < 5; i++)
                ttOb.tick(true);
            ttOb.tick(false);
        } else {
            for (int i = 0; i < 5; i++)
                ttOb.tock(true);
            ttOb.tock(false);
        }
    }
}
```

Output :

PS D:\MCA\MCA SEM 3\JAVA> java .\ThreadCom.java
Tick
Tock
Tick
Tock
Tick

Tock
Tick
Tock
Tick
Tock

Q. Suspend.java

```
class Suspend {
    public static void main(String[] args) {
        MyThread ob1 = new MyThread("My Thread");
        try {
            Thread.sleep(1000);

            ob1.mysuspend();
            System.out.println("Suspending thread.");
            Thread.sleep(1000);

            ob1.myresume();
            System.out.println("Resuming thread.");
            Thread.sleep(1000);

            ob1.mysuspend();
            System.out.println("Suspending thread.");
            Thread.sleep(1000);

            ob1.myresume();
            System.out.println("Resuming thread.");
            Thread.sleep(1000);

            ob1.mysuspend();
            System.out.println("Stopping thread.");
            ob1.mystop();
        } catch (InterruptedException e) {
            System.out.println("Main thread interrupted.");

        }
    }
}

class MyThread implements Runnable {
    Thread thrd;

    boolean suspended;
    boolean stopped;

    MyThread(String name) {
        thrd = new Thread(this, name);
        suspended = false;
        stopped = false;
        thrd.start();
    }

    // this is the entry point for thread
```

```
    public void run() {
        System.out.println(thrd.getName() + " Starting.");
        try {
            for (int i = 1; i < 1000; i++) {
                System.out.println(i + " ");
                if ((i % 10) == 0) {
                    System.out.println();
                    Thread.sleep(250);
                }
                synchronized (this) {
                    while (suspended) {
                        wait();
                    }
                    if (stopped)
                        break;
                }
            }
        } catch (InterruptedException e) {
            System.out.println(thrd.getName() + " Interrupted.");
        }
        System.out.println(thrd.getName() + " exiting.");

    }

    synchronized void mystop() {
        stopped = true;
        suspended = false;
        notify();
    }

    synchronized void mysuspend() {
        suspended = true;
    }

    synchronized void myresume() {
        suspended = false;
        notify();
    }
}
```

Output :

PS D:\MCA\MCA SEM 3\JAVA> java .\Suspend.java
My Thread Starting.
1
2

3
4
5
6
7
8
9
10

11
12
13
14
15
16
17
18
19
20

21
22
23
24
25
26
27
28
29
30

31
32
33
34
35
36
37
38
39

40

Suspending thread.
Resuming thread.
41
42
43
44
45
46
47
48
49
50

51
52
53
54
55
56
57
58
59
60

61
62
63
64
65
66
67
68
69
70

71
72
73

74
75
76
77
78
79
80

Suspending thread.
Resuming thread.
81
82
83
84
85
86
87
88
89
90

91
92
93
94
95
96
97
98
99
100

101
102
103
104
105
106
107
108

109
110

Stopping thread.
My Thread exiting.

Exercise - 3

Q. Write a program that, using an integer array of date values containing month, day, and year as integers for some number of dates (10, say, so the integer array will be two-dimensional with 10 rows and 3 columns), will write a file with a string representation of each date written as Unicode characters. For example, the date values 3,2,1990 would be written to the file as 2nd March 1990. Make sure that the date strings can be read back, either by using a separator character of some kind to mark the end of each string or by writing the length of each string before you write the string itself.

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class DateReadWrite {
    public static void main(String[] args) throws IOException {
        int[][] dates = { { 21, 1, 1998 }, { 1, 2, 1999 }, { 12, 3, 2000 }, {
26, 4, 2001 }, { 13, 5, 2002 },
                { 2, 6, 2003 }, { 5, 7, 2004 }, { 10, 8, 2005 }, { 20, 9, 2006
 }, { 31, 10, 2007 }, };

        FileWriter writer = new FileWriter("dates.txt");

        for (int i = 0; i < dates.length; i++) {
            Date d = new Date(dates[i][0], dates[i][1], dates[i][2]);
            String date = d.toString();
            System.out.println("Written: " + date);
            writer.write(date + "\n");
        }

        System.out.println("\nReading Data: ");
        writer.close();

        BufferedReader reader = new BufferedReader(new FileReader("dates.txt")
);
        String date;
```

```
        while ((date = reader.readLine()) != null) {
            System.out.println("Read: " + Date.getDate(date));
        }

        reader.close();
    }
}

class Date {
    private static final String[] months = { "January", "February", "March", "
April", "May", "June", "July", "August",
            "September", "October", "November", "December" };

    int dd, mm, yy;

    Date(int dd, int mm, int yy) {
        this.dd = dd;
        this.mm = mm;
        this.yy = yy;
    }

    private static String getDate(int dd) {
        String suffix = "th";
        if (dd == 1 || dd == 21 || dd == 31)
            suffix = "st";
        else if (dd == 2 || dd == 22)
            suffix = "nd";
        else if (dd == 3 || dd == 23)
            suffix = "rd";
        return dd + suffix;
    }

    private String getMonth(int mm) {
        return months[mm - 1];
    }

    private static int getMonth(String month) {
        switch (month) {
            case "January":
                return 1;
            case "February":
                return 2;
            case "March":
                return 3;
            case "April":
                return 4;
            case "May":
                return 5;
```

```
            case "June":
                return 6;
            case "July":
                return 7;
            case "August":
                return 8;
            case "September":
                return 9;
            case "October":
                return 10;
            case "November":
                return 11;
            default:
                return 12;
        }
    }

    static Date getDate(String date) {
        String[] values = date.split("[-]");
        int dd = Integer.parseInt(values[0].substring(0, values[0].length() -
2));
        int mm = getMonth(values[1]);
        int yy = Integer.parseInt(values[2]);
        return new Date(dd, mm, yy);
    }

    public String toString() {
        return (getDate(this.dd) + "-" + getMonth(this.mm) + "-" + this.yy);
    }
}
```

Output :

PS D:\MCA\MCA SEM 3\JAVA> java .\DateReadWrite.java
Written: 21st-January-1998
Written: 1st-February-1999
Written: 12th-March-2000
Written: 26th-April-2001
Written: 13th-May-2002
Written: 2nd-June-2003
Written: 5th-July-2004
Written: 10th-August-2005
Written: 20th-September-2006

Written: 31st-October-2007

Reading Data:
Read: 21st-January-1998
Read: 1st-February-1999
Read: 12th-March-2000
Read: 26th-April-2001
Read: 13th-May-2002
Read: 2nd-June-2003
Read: 5th-July-2004
Read: 10th-August-2005
Read: 20th-September-2006
Read: 31st-October-2007

Q. Extend the previous example to write a second file at the same time as the first, but containing the month, day, and year values as binary data. You should have both files open and be writing to both at the same time.

```
import java.io.*;

class SyncDateReadWrite {
    public static void main(String[] args) throws IOException {
        int[][] dates = { { 21, 1, 1998 }, { 1, 2, 1999 }, { 12, 3, 2000 }, { 
26, 4, 2001 }, { 13, 5, 2002 },
                { 2, 6, 2003 }, { 5, 7, 2004 }, { 10, 8, 2005 }, { 20, 9, 2006
 }, { 31, 10, 2007 }, };

        FileWriter writer = new FileWriter("dates.txt");
        DataOutputStream os = new DataOutputStream(new FileOutputStream("binar
y.dat"));

        for (int i = 0; i < dates.length; i++) {
            Date d = new Date(dates[i][0], dates[i][1], dates[i][2]);
            os.writeInt(dates[i][0]);
            os.writeInt(dates[i][1]);
            os.writeInt(dates[i][2]);
            String date = d.toString();
            System.out.println("Written: " + date);
            writer.write(date + "\n");
        }

        System.out.println("\nReading Data: ");
        writer.close();
        os.close();

        BufferedReader reader = new BufferedReader(new FileReader("dates.txt")
);
        DataInputStream is = new DataInputStream(new FileInputStream("binary.d
at"));
        String date;

        while ((date = reader.readLine()) != null) {
            System.out.println("Read: " + Date.getDate(date));
        }
        System.out.println("\nReading binary file: \n ");

        int datesRead[][] = new int[20][];
        int j = 0;
        while (is.available() > 0) {
            int dd = is.readInt();
            int mm = is.readInt();
            int yy = is.readInt();
```

```
            datesRead[j] = new int[3];
            datesRead[j][0] = dd;
            datesRead[j][1] = mm;
            datesRead[j][2] = yy;
            j++;

            System.out.print(dd + " - ");
            System.out.print(mm + " - ");
            System.out.println(yy);
        }

        reader.close();
        is.close();
    }
}

class Date {
    private static final String[] months = { "January", "February", "March", "
April", "May", "June", "July", "August",
            "September", "October", "November", "December" };

    int dd, mm, yy;

    Date(int dd, int mm, int yy) {
        this.dd = dd;
        this.mm = mm;
        this.yy = yy;
    }

    private static String getDate(int dd) {
        String suffix = "th";
        if (dd == 1 || dd == 21 || dd == 31)
            suffix = "st";
        else if (dd == 2 || dd == 22)
            suffix = "nd";
        else if (dd == 3 || dd == 23)
            suffix = "rd";
        return dd + suffix;
    }

    private String getMonth(int mm) {
        return months[mm - 1];
    }

    private static int getMonth(String month) {
        switch (month) {
            case "January":
                return 1;
```

```
            case "February":
                return 2;
            case "March":
                return 3;
            case "April":
                return 4;
            case "May":
                return 5;
            case "June":
                return 6;
            case "July":
                return 7;
            case "August":
                return 8;
            case "September":
                return 9;
            case "October":
                return 10;
            case "November":
                return 11;
            default:
                return 12;
        }
    }

    static Date getDate(String date) {
        String[] values = date.split("[-]");
        int dd = Integer.parseInt(values[0].substring(0, values[0].length() -
2));
        int mm = getMonth(values[1]);
        int yy = Integer.parseInt(values[2]);
        return new Date(dd, mm, yy);
    }

    public String toString() {
        return (getDate(this.dd) + "-" + getMonth(this.mm) + "-" + this.yy);
    }
}
```

Output :

PS D:\MCA\MCA SEM 3\JAVA> java .\SyncDateReadWrite.java
Written: 21st-January-1998
Written: 1st-February-1999
Written: 12th-March-2000

Written: 26th-April-2001
Written: 13th-May-2002
Written: 2nd-June-2003
Written: 5th-July-2004
Written: 10th-August-2005
Written: 20th-September-2006
Written: 31st-October-2007

Reading Data:
Read: 21st-January-1998
Read: 1st-February-1999
Read: 12th-March-2000
Read: 26th-April-2001
Read: 13th-May-2002
Read: 2nd-June-2003
Read: 5th-July-2004
Read: 10th-August-2005
Read: 20th-September-2006
Read: 31st-October-2007

Reading binary file:

21 - 1 - 1998
1 - 2 - 1999
12 - 3 - 2000
26 - 4 - 2001
13 - 5 - 2002
2 - 6 - 2003
5 - 7 - 2004
10 - 8 - 2005
20 - 9 - 2006
31 - 10 – 2007

Q. Write a program that, for a given String object defined in the code, will write strings to a file in the local character encoding (as bytes) corresponding to all possible permutations of the words in the string. For example, for the string the fat cat, you would write the strings the fat cat, the cat fat, cat the fat, cat fat the, fat the cat, and fat cat the, to the file, although not necessarily in that sequence. (Don't use very long strings; with n words in the string, the number of permutations is n!).

```
import java.io.*;

public class Permutation {
    public static void main(String[] args) throws IOException {
        String str = "the fat cat";
        String[] words = str.split("[ ]");

        FileWriter writer = new FileWriter("permutations.txt");

        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                for (int k = 0; k < 3; k++) {
                    String data = words[i] + words[j] + words[k] + "\n";
                    writer.write(data);
                }
            }
        }

        writer.close();

        BufferedReader reader = new BufferedReader(new FileReader("permutation
s.txt"));

        while ((str = reader.readLine()) != null)
            System.out.println(str);

        reader.close();

    }
}
```

Output :

PS D:\MCA\MCA SEM 3\JAVA> java .\Permutation.java

thethethe

thethefat

thethecat

thefatthe

thefatfat

thefatcat

thecatthe

thecatfat

thecatcat

fatthethe

fatthefat

fatthecat

fatfatthe

fatfatfat

fatfatcat

fatcatthe

fatcatfat

fatcatcat

catthethe

catthefat

catthecat

catfatthe

catfatfat

catfatcat

catcatthe

catcatfat

catcatcat