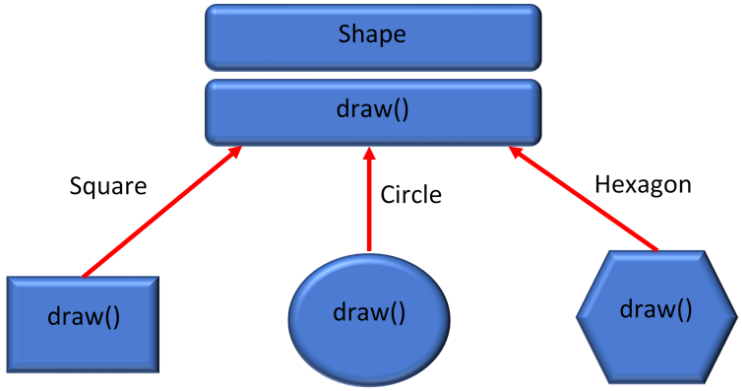
Exercise Solutions
Day 2

Exercise 1:




```
//Base class
class Shape
{
   void draw()
   {
      System.out.println("Inside the method of Parent class ");
      System.out.println("Drawing Shapes");
   }
}

//Derived class
class Circle extends Shape
{
   //Overriding method of base class with different implementation
   @Override
   void draw()
   {
      System.out.println("Inside the overridden method of the child class ");
      System.out.println("Drawing Circle");
   }
}

class Square extends Shape
{
   //Overriding method of base class with different implementation
   @Override
   void draw()
   {
      System.out.println("Inside the overridden method of the child class ");
      System.out.println("Drawing Square");
   }
```

```
}

class Hexagon extends Shape
{
   //Overriding method of base class with different implementation
   @Override
   void draw()
   {
      System.out.println("Inside the overridden method of the child class ");
      System.out.println("Drawing Hexagon");
   }
}
//Driver class
public class MethodOverridingDemo
{
   public static void main(String args[])
   {
      //creating object of Base class Shape
      // If a Parent type reference refers
      // to a Parent object, then Parent's draw() method is called

      Shape obj = new Shape();
      obj.draw();

      // If a Parent type reference refers to a Child object Child's draw() method is called.
      //This is called RUN TIME POLYMORPHISM.
      Shape obj1=new Circle();
      obj1.draw();

      Shape obj2 = new Square();
      obj2.draw();

      Shape obj3 = new Hexagon();
      obj3.draw();
   }
}
```

Exercise 2:
Create a class with a method that prints "This is parent class" and its subclass with another method that prints "This is child class". Now, create an object for each of the class and call
1 - method of parent class by object of parent class
2 - method of child class by object of child class
3 - method of parent class by object of child class

```java
class Pclass{
  public void pmethod(){
    System.out.println("This is parent class");
  }
}

class Cclass extends Pclass{
  public void cmethod(){
    System.out.println("This is child class");
  }
}

class Ans{
  public static void main(String[] args){
    Pclass m = new Pclass();
    Cclass n = new Cclass();
    m.pmethod();
    n.cmethod();
    n.pmethod();
  }
}
```

Exercise 3:

Create a class named 'Member' having the following members:

Data members

1 - Name

2 - Age

3 - Phone number

4 - Address

5 - Salary

It also has a method named 'printSalary' which prints the salary of the members.

Two classes 'Employee' and 'Manager' inherits the 'Member' class. The 'Employee' and 'Manager' classes have data members 'specialization' and 'department' respectively. Now, assign name, age, phone number, address and salary to an employee and a manager by making an object of both of these classes and print the same.

```java
class Member{
  String name;
  int age;
  String number;
  String address;
  int salary;

  public void printSalary(){
    System.out.println(salary);
  }

}

class Employee extends Member{
  String specialization;
}

class Manager extends Member{
  String department;
}

class Ans{
  public static void main(String[] args){
    Employee e = new Employee();
    e.name = "xyz";
    e.age = 23;
    e.number = "986****";
    e.address = "xyzxyz";
    e.salary = 1231;
    e.specialization = "xyzxyz";

    Manager m = new Manager();
    //Same goes for Manager
  }
}
```

Exercise 4:
Abstract Classes & Interfaces

a. Create an abstract class pen with methods write () and refill () as abstract methods
b. Use the pen class from Q1 to create a concrete class fountain pen with additional method change Nib ()
c. Create a class monkey with jump ( ) and bite ( ) methods Create a class human whichinherits this monkey class and implements basicanimal interface with eat ( ) and sleep methods
d. Create a class telephone with ( ) , lift ( ) and disconnected ( ) methods as abstract methods create another class smart telephone and demonstrate polymorphism
e. Demonstrate polymorphism using using monkey class from Q3
f. Create an interface TVremote and use it to inherit another interface smart TVremote
g. Create a class TV which implements TVremote interface from Q6

```
package com.company;

abstract class Pen{
   abstract void write();
   abstract void refill();
}

class FountainPen extends Pen{
   void write(){
      System.out.println("Write");
   }
   void refill(){
      System.out.println("Refill");
   }
   void changeNib(){
      System.out.println("Changing the nib");
   }
}
class Monkey{
   void jump(){
      System.out.println("Jumping...");
   }
   void bite(){
      System.out.println("Biting...");
   }
}

interface BasicAnimal{
   void eat();
   void sleep();
}
```

```
class Human extends Monkey implements BasicAnimal{
    void speak(){
        System.out.println("Hello sir!");
    }

    @Override
    public void eat() {
        System.out.println("Eating");
    }

    @Override
    public void sleep() {
        System.out.println("Sleeping");
    }
}

public class AbstractInterface {
    public static void main(String[] args) {
        // Q1 + Q2
        FountainPen pen = new FountainPen();
        pen.changeNib();

        // Q3
        Human harry = new Human();
        harry.sleep();

        // Q5
        Monkey m1 = new Human();
        m1.jump();
        m1.bite();
        // m1.speak(); --> Cannot use speak method because the reference is monkey which does
not have speak method

        BasicAnimal lovish = new Human();
        // lovish.speak(); --> error
        lovish.eat();
        lovish.sleep();

    }
}
```

Day 3

Exercise 1: Java program to demonstrate working of split() which splits with ":"
String: Quote: The Quick brown fox jumps over a lazy dog

```
public class Sample {
```

```
   public static void main(String[] args) {
      String str = "Quote: The Quick brown fox jumps over a lazy dog";
      String[] arrOfStr = str.split(":");

      for (String a : arrOfStr)
         System.out.println(a);
   }
}
```

Exercise 2: Write a program to find the length of the string "refrigerator".

```
class Ans{
  public static void main(String[] args){
    String a = "refrigerator";
    System.out.println(a.length());
  }
}
```

Exercise 3: Write a String concatenation program
String : Java Programming
Concatenate with: Language

```
public class Sample {
   public static void main(String[] args) {
      String s = "Java Programming ";
      s = s.concat("Language");
      System.out.print(s);
   }
}
```

Exercise 4: Write a java program to remove a particular character from a string ?
String: this is Java

```
public class Sample {
   public static void main(String[] args) {
      String str = "this is Java";
      System.out.println(removeCharAt(str, 3));
   }
   public static String removeCharAt(String s, int pos) {
      return s.substring(0, pos) + s.substring(pos + 1);
   }
   }
```

Exercise 5: Create a class that contains int, long, float and double fields. Create a toString( ) method for this class that uses String.format( ), and demonstrate that your class works correctly.

```
package com.company;

public class Main {
   private static final int i=100;
   private static final long l=10000l;
   private static final float f=10000.00f;
   private static final double d=100000.00;
```

```
  public String toString(){
    return String.format("Int: %1$-15d Long: %2$-15d Float: %3$-15.1f Double: %4$-15.7e", i, l, f,
d);
  }
  public static void main(String[] args){
    Main ex=new Main();
    System.out.println(ex);
  }
}
```

Exercise 6: Using the java.util.regex.Pattern, write a program to split the string "This!!unusual use!!of exclamation!!points".

```
package com.company;

import java.util.Arrays;
import java.util.regex.*;
import java.util.*;
public class Main {
  private static final String phrase = "This!!unusual use!!of exclamation!!points";
  public static void split(String regex){
    System.out.println(Arrays.toString(phrase.split(regex)));
  }
  public static void splitThreePieces(String regex){
    System.out.println(Arrays.toString(phrase.split(regex,3)));
  }
  public static void main(String[] args) {
    String regex = "!!";
    Main.split(regex);
    Main.splitThreePieces(regex);
  }
}
```

Exercise 7: Take 10 integer inputs from user and store them in an array and print them on screen.

```
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    int[] z = new int[10];
    for (int i = 0; i < z.length; i++) {
      System.out.println("Print the value of z[" + i + "]");
      z[i] = s.nextInt();
    }
    for (int i = 0; i < z.length; i++) {
      System.out.println("The value of z[" + i + "] is " + z[i]);
    }
  }
```

```
}
```

Exercise 8: Take 10 integer inputs from user and store them in an array. Now, copy all the elements in an another array but in reverse order.

```
package com.company;

import java.util.Scanner;

public class Main {
    public static void main(String[] args){
        Scanner s = new Scanner(System.in);
        int[] a = new int[10];
        int[] b = new int[10];
        for(int i =0;i<a.length;i++){
            System.out.println("Enter the value of a["+i+"]");
            a[i] = s.nextInt();
        }
        int j = 0;
        for(int i = b.length-1;i>=0;i--){
            b[i] = a[j];
            j++;
        }
        for(int i = 0; i< b.length; i++){
            System.out.println("The value of b["+i+"] is "+b[i]);
        }
       }
   }
```

Exercise 9: Take 20 integer inputs from user and print the following:
number of positive numbers
number of negative numbers
number of odd numbers
number of even numbers
number of 0s.

```
package com.company;

import java.util.Scanner;

public class Main {
    public static void main(String[] args){
         Scanner s = new Scanner(System.in);
         int[] z = new int[20];
         int pos = 0;
         int neg = 0;
```

```
        int odd = 0;
        int even = 0;
        int zero = 0;
        for(int i = 0;i<z.length;i++){
           System.out.println("Print the value of z["+i+"]");
           z[i] = s.nextInt();

           if(z[i]>0){
              pos++;
           }
           else if(z[i]<0){
              neg++;
           }
           else{
              zero++;
           }
           if(z[i]%2==0){
              even++;
           }
           else{
              odd++;
           }
        }
        System.out.println("Positive : "+pos+"\nNegative : "+neg+"\nZero : "+zero+"\nodd :
"+odd+"\neven : "+even);
     }
  }
```

Exercise 10: Take 10 integer inputs from user and store them in an array and Find largest and smallest elements of an array.

```
package com.company;

import java.util.Scanner;

public class Main {
    public static void main(String[] args){
       Scanner s = new Scanner(System.in);
       int[] a = new int[10];
       for(int i =0;i<a.length;i++){
          System.out.println("Enter the value of a["+i+"]");
          a[i] = s.nextInt();
       }

       int largest = a[0];
       int smallest = a[0];

       for(int i = 0;i<a.length;i++){
```

```
        if(a[i]>largest)
            largest = a[i];
        if(a[i]<smallest)
            smallest = a[i];
      }

      System.out.println("Largest is "+largest+" and smallest is "+smallest);
     }
  }
```

Exercise 11: Write a program to check if elements of an array are same or not it read from front or back. E.g.- 2,3,15,15,3,2 -> evaluates to true
2,3,15,15,2,1 -> evaluates to false

```
package com.company;

public class Main {
    public static void main(String[] args){
        int[] a = {2,3,15,15,3,2};
        boolean read = true;
        int j = a.length-1;

        for(int i =0;i<a.length/2;i++){
            if(a[i]!=a[j]){
                read = false;
                break;
            }
            else
                j--;
        }
        System.out.println(read);
       }
  }
```

Exercise 12: Write a method that creates and initializes a twodimensional array of double. The size of the array is determined by the arguments of the method, and the initialization values are a range determined by beginning and ending values that are also arguments of the method. Create a second method that will print the array generated by the first method. In main( ) test the methods by creating and printing several different sizes of arrays.

```
package com.company;

import java.util.Arrays;
import java.util.Random;

public class Main {
   public static Double[][] makeDoubleArray(int x, int y, int min, int max) {
      Random rand = new Random();
```

```
        Double[][] d = new Double[x][y];
        for (int i = 0; i < x; i++) {
            for (int j = 0; j < y; j++) {
                d[i][j] = (double) min + (max - min) * rand.nextDouble();
            }
        }
        return d;
    }

    public static void printArray(Double[][] d) {
        System.out.println(Arrays.deepToString(d));
    }

    public static void main(String[] args) {
        printArray(Main.makeDoubleArray(2, 3, 0, 100));
    }
}
```

Day 4

Exercise 1:

Exercise steps:

- Create a Java package called packageDemo .
- Inside the packageDemo package, create another package (subpackage) called employee
- Create a Java class called Employee inside the java package.
- Insert a main() method inside the Employee class.
- Inside the main() method, insert this statement:
- System.out.println("Employee Class executed");
- Compile and run the main() method of the Employee class.

```
package packageDemo.employee;

public class Employee {

    public static void main(String[] args) {
        System.out.println("Employee Class executed");
    }
}
```

Exercise 2:

- Create a package called com.packageDemo and inside that a subpackage called java .
- In the java package create a class called DataObject .
- Inside the DataObject class create a public member variable called count of type int .
- In the java package create a class called Main.
- In the Main class insert a main() method.
- Inside the main() method create an array of DataObject.
- Create 3 DataObject instances and assign a reference to them to element 0,1 and 2 in the array.
- Inside the main() method create a sum variable of type int .

- Loop through the array and add all count member variable values of the DataObject instances to the sum variable.
- Inside the main() method, insert a System.out.println() statement that prints out the value of the sum variable.

DataObject.java

```
package com.packageDemo.java;

public class DataObject {
   public int count = 0;
}
```

Main.java

```
package com.packageDemo.java;

public class Main {
   public static void main(String[] args) {
      DataObject[] dataObjects = new DataObject[3];

      DataObject dataObject = new DataObject();
      dataObject.count = 5;
      dataObjects[0] = dataObject;

      dataObject = new DataObject();
      dataObject.count = 7;
      dataObjects[1] = dataObject;

      dataObject = new DataObject();
      dataObject.count = 9;
      dataObjects[2] = dataObject;

      int sum = 0;

      for(int i=0; i < dataObjects.length; i++){
         sum = sum + dataObjects[i].count;
      }

      System.out.println("Sum: " + sum);

   }
}
```

Exercise 3:

Create a class with a main( ) that throws an object of class Exception inside a try block. Give the constructor for Exception a String argument. Catch the exception inside a catch clause and print the String argument. Add a finally clause and print a message to prove you were there.

```
package com.company;

class Test
```

```
{
   public static void main(String[] args) {
      try {
         String msg = "The quick brown fox jumps over a lazy dog!";
         throw new Exception(msg);
      } catch (Exception e) {
         System.out.println(e.getMessage());
      } finally {
         System.out.println("In finally block");
      }
   }
}
```

Exercise 4:
Define an object reference and initialize it to null. Try to call a method through this reference. Now wrap the code in a try-catch clause to catch the exception.

```
package com.company;

class Test
{
   public void foo() {System.out.println("The quick brown fox jumps over a lazy dog!");}
   public static void main(String[] args) {
      try {
        // Test test = new Test();
         Test test = null;
         test.foo();
      } catch(Exception e) {
         System.out.println(e);
      }
   }
}
```

Exercise 5:
Write code to generate and catch an ArrayIndexOutOfBoundsException.

```
package com.company;

class Test
{
   public static void main(String[] args) {
      try {
         int arrayLength = 5;
         int[] arrayToOverflow = new int[arrayLength];
         int loopTimes = 10;
         for (int i=0; i<loopTimes; i++) {
            System.out.println(arrayToOverflow[i] = i);
         }
      } catch (ArrayIndexOutOfBoundsException e) {
         System.out.println(e);
      }
```

```
    }
}
```

Exercise 6:
Create your own exception class using the extends keyword. Write a constructor for this class that takes a String argument and stores it inside the object with a String reference. Write a method that displays the stored String. Create a try-catch clause to exercise your new exception.

```
package com.company;

class Test
{
    public static class TestException extends Exception {
        private static final long serialVersionUID = 0l;
        private String message = "NULL";
        public TestException(){
            super();
        }
        public TestException(String msg) {
            message = msg;
        }
        public void showMessage() {
            System.out.println(message);
        }
    }
    public static void main(String[] args) {
        try {
            String justAMessage = "The quick brown fox jumps over a lazy dog!";
            throw new Test.TestException(justAMessage);
        } catch (Test.TestException e) {
            e.showMessage();
        }
    }
}
```

Exercise 7:
Create three new types of exceptions. Write a class with a method that throws all three. In main( ), call the method but only use a single catch clause that will catch all three types of exceptions.

```
package com.company;
class Test
{
    public static class TestException extends Exception {
        private static final long serialVersionUID = 0;
        private String message = "I am Exception 1 of Test!";
        public TestException() {
            super();
        }
        public TestException(String msg) {
            message = msg;
        }
```

```
    public String getMessage() {
      return message;
    }
  }
  public static class TestException2 extends Exception {
    private static final long serialVersionUID = 0;
    private String message = "I am Exception 2 of Test!";
    public TestException2() {
      super();
    }
    public TestException2(String msg) {
      message = msg;
    }
    public String getMessage() {
      return message;
    }
  }
  public static class TestException3 extends Exception {
    private static final long serialVersionUID = 0;
    private String message = "I am Exception 3 of Test!";
    public TestException3() {
      super();
    }
    public TestException3(String msg) {
      message = msg;
    }
    public String getMessage() {
      return message;
    }
  }
  public static void throwThreeException(int inNum) throws Exception{
    switch (inNum) {
      case 1: throw new Test.TestException();
      case 2: throw new Test.TestException2();
      case 3: throw new Test.TestException3();
      default: System.out.println("Congratulation! You win!");
    }
  }
  public static void main(String[] args) {
    int i = 1;
    while (true) {
      try {
        throwThreeException(i);
        break;
      } catch (Exception e) {
        System.out.println(e.getMessage());
        i++;
      }
    }
  }
}
```

Day 7

Exercise 1: Program to implement thread using runnable interface

```
class MyThread implements Runnable
{
    String message;
    MyThread(String msg)
    {
        message = msg;
    }
    public void run()
    {
        for(int count=0;count<=5;count++)
        {
            try
            {
                System.out.println("Run method: " + message);
                Thread.sleep(100);
            }
            catch (InterruptedException ie)
            {
                System.out.println("Exception in thread: "+ie.getMessage());
            }
        }
    }
}
```

Exercise 2: Program to creating multiple thread

```
class ThreadTest extends Thread
{
    private Thread thread;
    private String threadName;

    ThreadTest( String msg)
    {
        threadName = msg;
        System.out.println("Creating thread: " +  threadName );
    }
    public void run()
    {
        System.out.println("Running thread: " +  threadName );
        try
        {
            for(int i = 0; i < 5; i++)
            {
                System.out.println("Thread: " + threadName + ", " + i);
                Thread.sleep(50);
            }
        }
        catch (InterruptedException e)
        {
            System.out.println("Exception in thread: " +  threadName);
```

```
            }
            System.out.println("Thread " +  threadName + " continue...");
        }
        public void start ()
        {
            System.out.println("Start method " +  threadName );
            if (thread == null)
            {
                thread = new Thread (this, threadName);
                thread.start ();
            }
        }
}
public class MultipleThread
{
    public static void main(String args[])
    {
        ThreadTest thread1 = new ThreadTest( "First Thread");
        thread1.start();

        ThreadTest thread2 = new ThreadTest( "Second Thread");
        thread2.start();
    }
}
```

Exercise 3: Program to set priorities of thread

```
public class ThreadPriority extends Thread
{
    public void run()
    {
        System.out.println("run() method");
        String threadName = Thread.currentThread().getName();
        Integer threadPrio = Thread.currentThread().getPriority();
        System.out.println(threadName + " has priority " + threadPrio);
    }
    public static void main(String[] args) throws InterruptedException
    {
        ThreadPriority t1 = new ThreadPriority();
        ThreadPriority t2 = new ThreadPriority();
        ThreadPriority t3 = new ThreadPriority();

        t1.setPriority(Thread.MAX_PRIORITY);
        t2.setPriority(Thread.MIN_PRIORITY);
        t3.setPriority(Thread.NORM_PRIORITY);

        t1.start();
        t2.start();
        t3.start();
    }
}
```

Exercise 4: Program to display all running thread

```
class RunningThread extends Thread
{
    public static void main(String[] args)
    {
        System.out.println("Main methods");
        RunningThread  obj = new RunningThread();
        //set the thread name2
        obj.setName("\nThreadName ");
        //calling run () method
        obj.start();

        ThreadGroup currentGroup = Thread.currentThread().getThreadGroup();
        int numThreads = currentGroup.activeCount();
        Thread[] lstThreads = new Thread[numThreads];
        currentGroup.enumerate(lstThreads);
        //System.out.println(currentThread);
        for (int i =1;i< numThreads ; i++)
        {
            System.out.println("Number of thread: "+i + "  " + lstThreads[i].getName());
        }

        //checking the current running thread
        Thread currentThread = Thread.currentThread();
        System.out.println("Current running thread: "+currentThread);
    }
}
```

Exercise 5: Program for Synchronization block

SynchTest.java

```
class SynchTest
{
    void printNumber(int n)
    {
        synchronized(this)
        {
            System.out.println("Table of "+n);
            System.out.println("==========");
            for(int i=1;i<=10;i++)
            {
                System.out.println(n+" * "+i+" = "+(n*i));
                try
                {
                    Thread.sleep(400);
                }
                catch(Exception e)
                {
                    System.out.println(e);
                }
            }
```

```
        }
    }
}
```

MyThread1.java

```
class MyThread1 extends Thread
{
    SynchTest t;
    MyThread1(SynchTest t)
    {
        this.t=t;
    }
    public void run()
    {
        t.printNumber(7);
    }
}
```

SynchronizedBlock.java

```
public class SynchronizedBlock
{
    public static void main(String args[])
    {
        SynchTest obj = new SynchTest();
        MyThread1 t1=new MyThread1(obj);
        t1.start();
    }
}
```

Exercise 6: Write a Java program using Synchronized Threads, which demonstrates Producer Consumer concept.

```
public class ProducerConsumer
{
    public static void main(String[] args)
    {
        Shop c = new Shop();
        Producer p1 = new Producer(c, 1);
        Consumer c1 = new Consumer(c, 1);
        p1.start();
        c1.start();
    }
}
class Shop
{
    private int materials;
    private boolean available = false;
    public synchronized int get()
    {
```

```
        while (available == false)
        {
            try
            {
                wait();
            }
            catch (InterruptedException ie)
            {
            }
        }
        available = false;
        notifyAll();
        return materials;
    }
    public synchronized void put(int value)
    {
        while (available == true)
        {
            try
            {
                wait();
            }
            catch (InterruptedException ie)
            {
                ie.printStackTrace();
            }
        }
        materials = value;
        available = true;
        notifyAll();
    }
}
class Consumer extends Thread
{
    private Shop Shop;
    private int number;
    public Consumer(Shop c, int number)
    {
        Shop = c;
        this.number = number;
    }
    public void run()
    {
        int value = 0;
        for (int i = 0; i < 10; i++)
        {
            value = Shop.get();
            System.out.println("Consumed value " + this.number+ " got: " + value);
        }
    }
}
```

```
class Producer extends Thread
{
    private Shop Shop;
    private int number;

    public Producer(Shop c, int number)
    {
        Shop = c;
        this.number = number;
    }
    public void run()
    {
        for (int i = 0; i < 10; i++)
        {
            Shop.put(i);
            System.out.println("Produced value " + this.number+ " put: " + i);
            try
            {
                sleep((int)(Math.random() * 100));
            }
            catch (InterruptedException ie)
            {
                ie.printStackTrace();
            }
        }
    }
}
```

Exercise 7: Write a program that creates 2 threads - each displaying a message (Pass the message as a parameter to the constructor). The threads should display the messages continuously till the user presses ctrl+c.

Answer:

If any application threads (daemon or nondaemon) are still running at shutdown time, then the user just presses the ctrl+c to stop thread execution.

Thread1.java

```
class Thread1 extends Thread
{
   String msg = "";
   Thread1(String msg)
   {
      this.msg = msg;
   }
   public void run()
   {
      try
      {
         while (true)
         {
```

```
                System.out.println(msg);
                Thread.sleep(300);
            }
        }
        catch (Exception ex)
        {
            ex.printStackTrace();
        }
    }
}
```

Thread2.java

```
class Thread2 extends Thread
{
    String msg = "";
    Thread2(String msg)
    {
        this.msg = msg;
    }
    public void run()
    {
        try
        {
            while (true)
            {
                System.out.println(msg);
                Thread.sleep(400);
            }
        }
        catch (Exception ex)
        {
            ex.printStackTrace();
        }
    }
}
```

ThreadDemo.java

```
class ThreadDemo
{
    public static void main(String[] args)
    {
        Thread1 t1 = new Thread1("Running Thread1....");
        Thread1 t2 = new Thread1("Running Thread2....");
        t1.start();
        t2.start();
    }
}
```

Day 8

Exercise 1: Write a Java program to create a Date object using the Calendar class.

```
package com.dateTime;

import java.util.Calendar;

public class DateEx1 {
   public static void main(String[] args) {
        int year = 2016;
        int month = 0; // January
        int date = 1;

        Calendar cal = Calendar.getInstance();
        // Sets the given calendar field value and the time value
        // (millisecond offset from the Epoch) of this Calendar undefined.
        cal.clear();
        System.out.println();
        cal.set(Calendar.YEAR, year);
        cal.set(Calendar.MONTH, month);
        cal.set(Calendar.DATE, date);

        System.out.println(cal.getTime());
        System.out.println();
   }
}
```

Exercise 2: Write a Java program to get current full date and time.

```
package com.dateTime;

import java.util.Calendar;

public class DateEx2 {
   public static void main(String[] args) {
      Calendar now = Calendar.getInstance();
      System.out.println();
      System.out.println("Current full date and time is : " + (now.get(Calendar.MONTH) + 1) + "-"
            + now.get(Calendar.DATE) + "-" + now.get(Calendar.YEAR) + " "
            + now.get(Calendar.HOUR_OF_DAY) + ":" + now.get(Calendar.MINUTE) + ":"
            + now.get(Calendar.SECOND) + "." + now.get(Calendar.MILLISECOND));
      System.out.println();
   }
}
```

Exercise 3: Write a Java program to get the number of days of a month.

```
package com.dateTime;

import java.util.Calendar;

public class DateEx2 {
   public static void main(String[] args) {
```

```
        Calendar cal = Calendar.getInstance();
        int days = cal.getActualMaximum(Calendar.DAY_OF_MONTH);
        System.out.println();
        System.out.println("Number of days of the current month : " + days);
        System.out.println();
    }
}
```

Exercise 4: Write a Java program to get a day of the week of a specific date.

```
package com.dateTime;

import java.util.Calendar;

public class DateEx2 {
    public static void main(String[] args) {
        // Create a default calendar
        Calendar cal = Calendar.getInstance();
        //Set your date: cal.setTime(yourDate);
        System.out.println();
        int dayOfWeek = cal.get(Calendar.DAY_OF_WEEK);
        System.out.println("Day of the week : " + dayOfWeek);
        System.out.println();
    }
}
```

Exercise 5: Write and read a plain text file using FileWriter and BufferedReader.

```
package com.dateTime;

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

public class Exercise1 {
    public static void main(String[] args) {
        StringBuilder sb = new StringBuilder();
        String strLine = "";
        try
        {
            String filename= "D:\\myfile.txt";
            FileWriter fw = new FileWriter(filename,false);
            //appends the string to the file
            fw.write("Java File I/O Demo\n");
            fw.close();
            BufferedReader br = new BufferedReader(new FileReader("D:\\myfile.txt"));
            //read the file content
            System.out.println("The file content is: ");
            int r = 0;
            while ((r = br.read()) != -1) {
                System.out.print((char) r);
```

```
            }
            br.close();
        }
        catch(IOException ioe)
        {
            System.err.println("IOException: " + ioe.getMessage());
        }
    }
}
```