

B.M.S COLLEGE OF ENGINEERING BENGALURU

Autonomous Institute, Affiliated to VTU



LAB REPORT

23CS3PCOOJ

Submitted in partial fulfilment of the requirements for Lab
Bachelor of Engineering
in
Computer Science and Engineering

Submitted by:

PRAJWAL K K

(1BM22CS199)

Department of Computer Science and Engineering,
B.M.S College of Engineering,
Bull Temple Road, Basavanagudi, Bangalore, 560 019
2023-2024.

INDEX

Sl.No.	Title	Date
1	Complete scanned Observation Book	12/12/2023 - 20/02/2024
2	Lab 1	12/12/2023
3	Lab 2	19/12/2023
4	Lab 3	26/12/2023
5	Lab 4	02/01/2024
6	Lab 5	09/01/2024
7	Lab 6	16/01/2024
8	Lab 7	23/01/2024
9	Lab 8	30/01/2024
10	Lab 9	06/02/2024
11	Lab 10	20/02/2024

INDEX :- JAVA :-

NAME: PRAJWAL.K. STD: _____ SEC: _____ ROLL NO: _____

Q1. ParseInt:

Area of a Rectangle and Verify the Same with various inputs (length, breadth).

Sol:

```
class RectangleArea
```

```
{
```

```
public static void main(String args[])
```

```
{
```

```
int length, breadth;
```

```
length = Integer.parseInt(args[0]);
```

```
breadth = Integer.parseInt(args[1]);
```

```
int area = length * breadth;
```

```
System.out.println("Length of rectangle = " + length);
```

```
System.out.println("Breadth of rectangle = " + breadth);
```

```
System.out.println("Area of rectangle = " + area);
```

```
}
```

Q2. Scanner:

Sol:

```
import java.util.Scanner;
```

```
class HelloWorld
```

```
{
```

```
public static void main(String args[])
```

```
{
```

```
int a; float b; String s;
```

```
Scanner in = new Scanner(System.in);
```

```
System.out.println("Enter a string");
```

```
s = in.nextLine();
```

```
System.out.println("You entered string " + s);
```

```
System.out.println("Enter a integer");
```

```
a = in.nextInt();
```

```
System.out.println("You entered integer "+a);
System.out.println("Enter a float");
b=in.nextFloat();
System.out.println("You entered float "+b);
}
```

03. Factorial:-

```
import java.util.Scanner;
class factorial
{
    public static void main (String args[])
    {
        int fac=1;
        System.out.println("Enter a number:");
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        for (int i=1 ; i<=n; i++)
        {
            fac=fac*i;
        }
        System.out.println("The factorial : "+fac);
    }
}
```

04. Palindrome:-

```
import java.util.Scanner;
class palindrome
{
    public static void main (String args[])
    {
```

```

int n, t, rem, rev=0;
Scanner sc = new Scanner (System.in);
System.out.print ("Enter a 5 digit number: ");
n = sc.nextInt();
t = n;
while (t>0)
{
    rem = t%10;
    rev = rev*10 + rem;
    t = t/10;
}
if (rev==n)
    System.out.print ("Palindrome");
else
    System.out.print ("not palindrome");
    
```

Q5. Sum of digits in a 5 digit number.

import java.util.Scanner;

class sumofdigits

{

public static void main (String args[])
{

Long number, sum;

Scanner sc = new Scanner (System.in);

System.out.print ("Enter a 5 digit number: ");

number = sc.nextInt();

```
for (sum = 0; number != 0; number =  
    number / 10)  
{  
    sum = sum + number % 10;  
}  
System.out.println ("Sum of digits :: " + sum);  
}
```

06. Array:

```
class AutoArray  
{  
public static void main (String args [])  
{  
    int monthDays [] = {31, 28, 31, 30, 31, 30, 31,  
                        31, 30, 31, 30, 31};  
    System.out.println ("April has "  
                      + monthDays [3] + " days");  
}
```

07 Type Conversion:

```
class promote  
{  
public static void main (String args [])  
{  
    byte b = 42;  
    char c = 'a';  
    short s = 1024;  
    int i = 5000;
```

float f = 6.74f;
double d = 0.1234;

double result = (f * b) + (i / c) - (d * s);

System.out.println((f * b) + " + " + (i / c) + "
+ (d * s));

}

}

double result = (f * b) + (i / c) - (d * s);

(f * b) + (i / c) - (d * s) = 3.000000

(f * b) + (i / c) - (d * s) = 3.000000

3.000000

Pattern: e=0

(i) Pattern: e=d

(c) Pattern: e>d

e>d

Condition:

else (e <= d) {

System.out.println

(f * b) + (i / c) - (d * s);

(f * b) + (i / c) - (d * s) = 3.000000

(f * b) + (i / c) - (d * s) = 3.000000

(f * b) + (i / c) - (d * s) = 3.000000

(f * b) + (i / c) - (d * s) = 3.000000

$\star \Rightarrow$ quadratic equation $ax^2 + bx + c = 0$
 import java.util.Scanner;
 class quadratic
 {
 int a, b, c;
 double r1, r2, d;
 void getd()
 {
 Scanner s = new Scanner(System.in);
 System.out.print("Enter the coefficients
 of a, b, c");
 a = s.nextInt();
 b = s.nextInt();
 c = s.nextInt();
 }
 void computd()
 {
 while (a == 0)
 {
 System.out.print("Not a
 quadratic equation");
 System.out.print("Enter a non zero value of a");
 }
 Scanner s = new Scanner(System.in);
 a = s.nextInt();
 d = b * b - 4 * a * c;
 if (d == 0)
 }
 }

$$r_1 = (-b) / (2 \cdot a);$$

System.out.println ("Roots are real & equal");

System.out.println ("Root 1 = Root 2 = " + r1);

}

else if (d > 0)

{

$$r_1 = ((-b) + (\text{Math.sqrt}(d))) / (2 \cdot a);$$

$$r_2 = ((-b) - (\text{Math.sqrt}(d))) / (2 \cdot a);$$

System.out.println ("Roots are real & distinct");

System.out.println ("Root 1 = " + r1 + "Root 2 = " + r2);

}

else if (d < 0)

{

System.out.println ("Roots are imaginary");

$$r_1 = (-b) / (2 \cdot a);$$

$$r_2 = \text{Math.sqrt}(-d) / (2 \cdot a);$$

System.out.println ("Root 1 = " + r1 + " + i " + r2);

System.out.println ("Root 1 = " + r1 + " - i " + r2);

}

}

class Quadratic

{

public static void main (String args[])

{

Quadratic q = new Quadratic();

q.getd();

q.compute();

}

→ Output :-

(i) Enter the coefficients of a,b,c:

0 6 3

Roots are real and distinct

Root₁ = Root₂ = -1

(ii) Enter the coefficients of a,b,c:

0 2 3

Not a quadratic equation

Enter a non zero value of a

(iii) Enter the coefficients of a,b,c:

1 2 1

Roots are real and equal

Root₁ = Root₂ = -1

(iv) Enter the coefficients of a,b,c:

1 -3 2

Roots are real & distinct

Root₁ = 2 Root₂ = 1

(v) Enter the coefficients of a,b,c:

1 1 2

Roots are imaginary

Root₁ = 0.0 + i 0.322875

Root₂ = 0.0 - i 0.322875

Q. Develop a Java program to create a class Student with members USN, name, an array credits and an array marks.

Sol:
import java.util.Scanner;
class Subject

int subjectMarks;
int Credits;
int grade;

Public class Student

{
Subject[] Subject;
String name;
String USN;
double SGPA;

Scanner S;

Student()
{

int i;

Subject = new Subject[9];

for (i=0; i<9; i++)

subject[i] = new Subject();

s = new Scanner(System.in);

}

public void getStudentDetails()

System.out.println("Enter Name:");

name = s.nextLine();

System.out.println("Enter USN:");

name = s.nextLine();

```
3
public void getMarks()
{
    for (int i=0; i<8; i++)
    {
        System.out.print("Enter marks for
Subject " + (i+1) + ": ");
        Subject[i].SubjectMarks = S.nextInt();
        System.out.print("Enter Credits for
Subject " + (i+1) + ": ");
        if (Subject[i].SubjectMarks >= 100)
        {
            System.out.println("Invalid marks
entered ");
            i--;
        }
        else if (Subject[i].SubjectMarks >= 90)
        {
            Subject[i].grade = 10;
        }
        else if (Subject[i].SubjectMarks >= 80)
        {
            Subject[i].grade = 9;
        }
        else if (Subject[i].SubjectMarks >= 70)
        {
            Subject[i].grade = 8;
        }
        else if (Subject[i].SubjectMarks >= 60)
        {
            Subject[i].grade = 7;
        }
    }
}
```

```
else if (Subject[i].SubjectMarks >= 50)
```

```
{ Subject[i].Grade = 6;
```

```
} else if (Subject[i].SubjectMarks >= 40)
```

```
{ Subject[i].Grade = 0;
```

```
}
```

```
public void computeSGPA()
```

```
{ double totalCredits = 0;
```

```
double totalGradePoints = 0;
```

```
for (int i = 0; i < 8; i++)
```

```
{
```

```
totalCredits += Subject[i].Credits;
```

```
totalGradePoints += Subject[i].Grade * Subject[i].Credits;
```

```
}
```

```
SGPA = totalGradePoints / totalCredits;
```

```
}
```

```
public static void main (String [] args)
```

```
{
```

```
Student s1 = new Student();
```

```
s1.getStudentDetails();
```

```
s1.getSubjectMarks();
```

```
s1.computeSGPA();
```

```
System.out.println ("Name : " + s1.name);
```

```
System.out.println ("UIN : " + s1.uin);
```

```
System.out.println ("SGPA : " + s1.SGPA);
```

```
}
```

Output:

Enter Name:

Pragyal KK

Enter USN:

IB0722CS199

Enter Marks for Subject 1:

89

Enter Credits for Subject 1:

4

Enter Marks for Subject 2:

86

Enter Credits for Subject 2:

4

Enter Marks for Subject 3:

79

Enter Credits for Subject 3:

4

Enter Credits for Subject 4:

90

Enter Credits for Subject 4:

3

Enter Marks for Subject 5:

93

Enter Credits for Subject 5:

3

Enter Marks for Subject 6:

91

Enter Credits for Subject 6:

3

Enter Credits for Subject 7:

87

Enter Credits for Subject 7:

2

Enter marks for subject 8:

Enter credits for subject 8:

Name: Praywal KK

USN: 1B1622CS99

SGPA: 9.28

19/12/23

26/12/2023

LAB-03:-

Q.

Create a class Book which contains four members : name, author, price, numPages. Include methods to set and get the details of the objects. Include a toString() method that could display the complete details of the book. Develop a Java program to create n book objects.

Sol:

```
import java.util.Scanner;
```

Class Book :

{

```
    String name;  
    String author;  
    int price;  
    int numPages;
```

```
Public Book (String name, String author,  
            int price, int numPages)  
{
```

```
    this.name = name;  
    this.author = author;  
    this.price = price;  
    this.numPages = numPages;
```

```
Public String toString()  
{
```

```
    String name, author, price, numPages;
```

```

    name = "Book name :" + this.name + "\n";
    author = "Author name :" + this.author + "\n";
    price = "Price :" + this.price + "\n";
    numPages = "Number of pages :" + this.numPages;
    return name + author + price + numPages;
}
}

```

public class Main

```

{
    public static void main (String [] args)
    {
        Scanner s = new Scanner (System.in);
        int n;
        String name;
        String author;
        int price;
        int numPages;
    }
}
```

System.out.println ("Enter the number of books :");

n = s.nextInt();

Book [] b = new Book [n];

```

for (int i=0; i<n; i++)
{
```

System.out.println ("Enter the name of the book :");

name = s.next();

System.out.println ("Enter the author of the book :");

author = s.next();

System.out.println("Enter the price of
the book: ");
price = s.nextInt();

System.out.println("Enter the number
of pages of the book: ");
numPages = s.nextInt();

b[i] = new Book (name, author, price,
numPages);

for (int i=0; i<n; i++)

System.out.println(b[i].toString());

Output:-

Enter the number of books: 2

Enter name of the book:

ABC

Enter author of the book:

Arnold

Enter the price of the book:

300

Enter the number of pages of the book:

450

Enter name of the book:

XYZ

Enter author of the book:

Ranil

Enter the price of the book:

500

Enter the number of pages of the book:

390

Book name: ABC

Author name: Arunind

Price: 300

Number of pages: 450

Book name: XYZ

Author name: Ranil

Price: 500

Number of pages: 390

For
26/12/23

02/01/2023

LAB-04:-

- Q) Develop a Java program to create an abstract class named Shape that contains two integers and an empty method named printArea(). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contain only the method printArea() that prints the area of the given shape.

Sol)

```
import java.util.Scanner;  
  
abstract class Shape  
{  
    int length;  
    int breadth;  
  
    abstract void printArea();  
}
```

class Rectangle extends Shape

```
{  
    void printArea()  
    {
```

```
        Scanner s = new Scanner(System.in);  
        System.out.println("Enter length and  
        breadth of rectangle : ");
```

```
        length = s.nextInt();
```

```
        breadth = s.nextInt();
```

```
        System.out.println("The Area of  
        Rectangle is "+(length * breadth));  
    }
```

3

```
class Triangle extends Shape
{
    void printArea()
    {
        Scanner S = new Scanner (System.in);
        System.out.println ("Enter the dimension
                            by Triangle : ");
        length = S.nextInt();
        breadth = S.nextInt();
        System.out.println ("The Area of
                            Triangle is " + (0.5 * length * breadth));
    }
}
```

~~Scanner algorithm for area~~

```
class Circle extends Shape
{
    void printArea()
    {
        Scanner S = new Scanner (System.in);
        System.out.println ("Enter the dimension
                            by Circle : ");
        length = S.nextInt();
        System.out.println ("The Area of Circle
                            is " + (3.14 * length * length));
    }
}
```

~~Scanner algorithm for area~~

```
public class Main
{
    public static void main (String args[])
    {
        shape shape;
    }
}
```

```
shape = new Rectangle();
shape.printArea();
shape = new Triangle();
shape.printArea();
shape = new Circle();
shape.printArea();
}
```

∴ Output:

Enter the dimensions of the Rectangle :

7 9

Area of rectangle is 63

Enter the dimensions of the Triangle :

5 8

Area of triangle is 20

Enter the dimension of the Circle :

9

Area of circle is 254.34

*Fin
02/01/24*

~~01/01/24~~ TAB "05" :-

Q Create a class Account that stores customer name, account number and type of account. From this derive the classes Current-account and Saving-account to make them more specific to their requirements. Include the necessary methods in order to achieve the following tasks:

(a) Accept deposit from customer and update the balance.

(b) Display the balance

(c) Compute and deposit interest

(d) Permit withdrawal and update the balance

* Check for the minimum balance, impose penalty if necessary and update the balance.

~~SOP~~

import java.util.Scanner;

class Account

{

String customerName;

int accountNumber;

String accountType;

double balance;

Account(String customerName, int accountNumber, String accountType, double balance)

this.customerName = customerName;

this.accountNumber = accountNumber;

this.accountType = accountType;

```
this.balance = balance; }  
void deposit(double amount) {  
    balance += amount;  
    System.out.println("Deposit of " + amount  
        + " successful");  
}
```

```
void displayBalance() {  
    System.out.println("Balance: " + balance);  
}
```

```
void withdraw(double amount) {  
    if (balance - amount < 0)  
        System.out.println("Insufficient balance");  
    else  
        balance -= amount;  
}
```

```
System.out.println("Withdrawal of "  
    + amount + " successful");  
}
```

```
class SavingAccount extends Account
```

```
{  
    String customerName,  
    int accountNumber, String accountType,  
    double balance }  
}
```

```
{
    Super (CustomerName, accountNumber, accountType,
           balance);
```

```
void compoundInterest()
```

```
{
```

```
    double rate = 0.05;
```

```
    double time = 1.0;
```

```
    double interest = balance * Math.pow(1 +
        rate, time) - balance;
```

```
    balance += interest;
```

```
System.out.println ("Interest of " + interest +
    " added");
```

```
}
```

```
void withdraw (double amount)
```

```
{
```

```
    if (balance - amount < 0)
```

```
{
```

```
        System.out.println ("Insufficient
```

```
        balance");
```

```
        return;
```

```
}
```

```
    balance -= amount;
```

```
    System.out.println ("withdrawal of "
```

```
+ amount + " successful");
```

```
}
```

```
class CurrentAccount extends Account
```

```
{
```

double minimumBalance = 1000;

double serviceCharge = 50;

CurrentAccount (String customerName,
int accountNumber, String accountType,
double balance)

{

Super (customerName, accountNumber,
accountType, balance);

}

void withdraw (double amount)

{

if (balance - amount < minimumBalance)

System.out.println ("Insufficient

balance");

return;

}

balance -= amount;

System.out.println ("withdrawal of "

+

amount + "Successful");

}

void imposeServiceCharge ()

{

if (balance < minimumBalance)

balance -= serviceCharges;

System.out.println ("Service

charge of " + serviceCharge + " imposed");

}

3

```
public class Bank
```

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

```
{  
    Scanner scanner = new Scanner (System.in);
```

```
    System.out.println ("Enter customer name: ");
```

```
    String customerName = scanner.nextLine ();
```

```
    System.out.println ("Enter account number: ");
```

```
    int accountNumber = scanner.nextInt ();
```

```
    System.out.println ("Enter account type  
    (Savings / Current): ");
```

```
    String accountType = scanner.next ();
```

```
    System.out.println ("Enter initial balance: ");
```

```
    double balance = scanner.nextDouble ();
```

```
    Account account;
```

```
    if (accountType.equals ("Savings"))
```

```
        account = new SavingsAccount
```

```
        (customerName, accountNumber,
```

```
        accountType, balance);
```

```
    else
```

```
        if
```

```
            account = new CurrentAccount
```

```
            (customerName, accountNumber,
```

```
            accountType, balance);
```

```
    while (true)
```

{

```
System.out.println ("1. Deposit");
System.out.println ("2. Display balance");
System.out.println ("3. Compute and
deposit interest");
System.out.println ("4. Withdrawal");
System.out.println ("5. Exit");
System.out.print ("Enter choice: ");
int choice = scanner.nextInt();
```

Switch (choice)

{

(Case 1:

```
System.out.println ("Enter amount to
be deposit");
```

```
double amount = scanner.nextDouble();
```

```
account.deposit (amount);
```

```
break;
```

(Case 2:

```
account.displayBalance ();
```

```
break;
```

Case 3:

```
if (account instanceof Saving Account)
```

```
((Saving Account) account).compound
Interest ();
```

}

else

{

```
System.out.println ("Interest not
available for current account");
```

break;

Case 4:

```
System.out.println("Enter amount to withdraw:");
amount = scanner.nextDouble();
account.withdraw(amount);
if (account instanceof CurrentAccount)
    ((CurrentAccount) account).imposeServiceCharge();
}
```

break;

Case 5:

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

}

}

}/.

-: Output :-

Enter customer name: Prajwal Kothari,

Enter account number: 199

Enter account type (savings/current): savings

Enter initial balance: 2890

1. Deposit

2. Display balance

3. Compute and deposit interest

4. Withdraw

5. Exit

Enter choice : 3

Interest of 144.5 added

1. Deposit
2. Display balance
3. Compute and deposit interest
4. Withdraw
5. Exit

Enter choice : 4

Enter amount to withdraw : 1770

withdraw of 1770.0 successful

1. Deposit
2. Display balance
3. Compute and deposit interest
4. withdraw
5. Exit

Enter choice : 2

Balance : 1264.5

1. Deposit
2. Display balance
3. Compute and deposit interest
4. withdraw
5. Exit

Enter choice : 5

11/01/24

LAB-'06' Strings :-

01. Demonstrate various string constructor with proper java program.

S019

:- Output:-

//String

Hello, man!

Java;

02. Demonstrate string length, string literal; string concat

S019

:- Output:-

length = 8

concatenated: Projwalk Programming

03. Demonstrate toString()

S019

:- Output:-

Student: Demo

04. Using getchars(), extract Bmsce from "Welcome to Bmsce college"

S019

:- Output:-

Extracted: Bmsce

Q5: Demonstrate `getBytes()`, `toCharArray()` with proper java programs

SOP

∴ Output :-

Bytes : Prajwal, K!

Chars : Prajwal, K!

Q6: Check the following output and write the java programs using string functions

Bmsre equals Bmsre -> true

Bmre equals college -> false

Bmre equals Bmsre -> false

Bmsre equals Ignoucas e Bmsre -> true

SOP

∴ Output :-

true

false

false

true

Q7: Using regularmatchers) find the substring "Bmsre College" from the string "Welcome to Bmsre College of Engineering". If matches display substring is matched otherwise display not matched

-: Output :-

Substring is matched.

08. Demonstrate startwith() to give output true and false (endswith())

Sol:

-: Output :-

true

false.

09. Demonstrate endswith() to give output true and false.

Sol:

-: Output :-

endswith()

false

10. Demonstrate a java program to show the output for equals() & equals ==

Sol:

equals () Verus ==

World equals world -> true

World == World -> false.

11. Write a java program to perform sorting for alphabets using Comparator()

sop Compare to()

apple

ball

cat

dog

free

gum

hen

ice

jug

kite

man

net

orange

parrot

quar

ring

star

yatch

ze

12 Write a java program to perform sorting of number from 10 to 1 using compareto.

sop

(comparator)

1

2

3

4

5

6

7

8

9

13. WAP Using `substring()`, `indexof()`, `+`, `for` replacing "was" to "is". This was was a test.
This was, too.

Sol:

`substring()`, `indexof()`
this was was a test. This was was, too.
This was a test. This was was, ~~was~~ too
This is a test. This was was, too
This is a test. This was, ~~was~~ too
This is a test. This is, too,

14. WAP to demonstrate `concat()` for `s1 = "Hello"` and `s2 = "World"`

Sol:

`Concat()`
`HelloWorld`

15. WAP to demonstrate `replace()`. Replace "College" with "Commege".

Sol:

`replace()`
`Commege`

16. WAP to demonstrate `trim()` for "Hello Friends".

Sol:

`trim()`
~~Hello Friends.~~

17. Design a class which represents a student. Every student record is made up of the following fields.

- i) Regist No (int)
- ii) Full Name (string)
- iii) Semester (short)
- iv) CGPA (float)

Sol:

Student Records for Student 1:

Registration Number : 8

Full Name: Praywalk

Semester: 2

(CGPA: 9.4)

Student Record:

Registration Number: 8

Full Name: Praywalk

Semester: 2

(CGPA: 9.4)

18. Demonstrate string buffer functions like setLength(), charAt(), setCharAt(), getCharAt(), append(), Insert(), reverse(), delete(), deleteCharAt(), Replace(), Substring(). with simple java programs

Sol:

String Buffer

After setLength(5): Hello

character at index 2: H

After setCharAt(0, 'x'): xello

getChar(0, 5): xello.

19. WAP to create an abstract class Bird with abstract methods fly() and makeSound(). Create subclss Eagle and Hawk that extends the Bird class and implement the respective methods to describe how each bird flies and makes a sound.

SOP: Inherit 3 levels with and return 12.

Eagle Actions:

Eagle is soaring high in the sky

Eagle Snatches (loudly)

Hawk Actions:

Hawk is gliding through the air

Hawk makes sharp cry

20. WAP to create an abstract class Shape with abstract methods calculateArea() and calculatePerimeter(). Create subclss Circle and Triangle that extends the shape class and implement the respective methods to calculate the area and perimeter of each shape.

SOP: Inherit 3 levels with and return 12.

Circle Area: 78.5398 ; Perimeter: 31.4159

Triangle Area: 6.0 ; Perimeter: 12.0

LAB-06:-

~~Q~~ Create a package CIE which has two classes Student and Internals. The class Student has members an array that stores the Internal marks scored in five courses of the current semester of the student. Create another package SEE which has the class External which is a derived class of Student. This class has an array that stores the SEE marks scored in five courses of the current semester of the student. Import the two packages in a file that declares the final marks of n students in all five courses.

SOL

1) Student.java

```
Package CIE;
import java.util.Scanner;
Public class Student
{
    Protected String USN = newString();
    Protected String name = newString();
    Protected int sem;
    Public void input Student Details()
    {
        Scanner s = new Scanner (System.in);
        System.out.println("Enter the USN");
        USN = s.nextLine();
        System.out.println("Enter the Student name");
    }
}
```

```

name = s.nextLine();
System.out.println("Enter the Semester");
Sem = s.nextInt();
}
    
```

Public void display Student Details()

```

{
System.out.println("USN = " + USN);
System.out.println("Student name = " + name);
System.out.println("Semester = " + SEM);
}
    
```

II Internals.java

```

Package CIE;
import CIE.Student;
import java.util.Scanner;
Public class Internals extends Student
{
    
```

```
Protected int marks[] = new int[5];
```

```
Public void input (IE marks)
{
    
```

```
Scanner S = new Scanner (System.in);
```

```
System.out.println("Enter the mark of
each subject ");
```

```
for (int i=0 ; i<5 ; i++)
    
```

```
marks[i] = S.nextInt();
```

```
}
```

```
}
```

11 External.java

Package SEE;
import CIE. Internals;

import java.util.Scanner;

Public class External extends Internals

Protected int marks [];

Protected int final Marks [];

Public External()

{

marks = new Int [5];

final Marks = new int [5];

Public void input SEE marks()

{

Scanner s = new Scanner (System.in);

for (int i = 0; i < 5; i++)

{

System.out.print ("Subject " + (i + 1) +
marks + " ");

marks [i] = s.nextInt();

}

Public void calculate Final marks()

for (int i = 0; i < 5; i++)

final Marks [i] = marks [i] / 2 +
Super. marks [i];

Public void display Final marks()

```

{
    display Student Details ();
    for (int i=0; i<5; i++)
        System.out.println ("Subject " + (i+1) +
": " + final Marks [i]);
}
}

```

```

import SEE.externals;
class Marks
{
    public static void main (String args[])
    {
        int num of Students = 2;
        Externals final Marks [] = new Externals [num of Students];
        for (int i=0; i<num of Student; i++)
        {
            final Marks [i] = new Externals ();
            final Marks [i].input Student Details ();
            System.out.println ("Enter I.E marks");
            final Marks [i].input (I.E marks ());
            System.out.println ("Enter S.E marks");
            final Marks [i].input SEE marks ();
        }
        System.out.println ("Displaying data:\n");
        for (int i=0; i<num of student; i++)
        {
            final Marks [i].calculate Final
            Marks ();
            final Marks [i].display Final marks ();
        }
    }
}

```

÷ Output:-

Enter the USN

199

Enter the Student Name

Prajwalk

Enter the Semester

3

Enter CIE marks

Enter the marks of each Subject:

48

49

47

35

50

Enter SEE marks:-

Subject 1 marks : 45

Subject 2 marks : 47

Subject 3 marks : 49

Subject 4 marks : 50

Subject 5 marks : 46

Enter the USN

211

Enter the student name

Ravi

Enter the Semester

3

Enter CIE marks.

Enter the marks of each subject

44

42

47

46

41

Enter SEE marks

Subject 1 marks : 49

Subject 2 marks : 44

Subject 3 marks : 47

Subject 4 marks : 50

Subject 5 marks : 42

Display Data:

USN = 1BEM22CS199

Student name = Prajwak

Semester = 3

Subject 1 : 70

Subject 2 : 72

Subject 3 : 71

Subject 4 : 60

Subject 5 : 73

USN = 211

Student name : Rauer

Semester : 3

Subject 1 : 93

Subject 2 : 86

Subject 3 : 94

Subject 4:96

Subject 5:83

✓

PP
SP
FP
O2
CP

exam 333 noted

PP : exam 1 trip.2

SP : exam 8 trip.2

FP : exam 8 trip.2

O2 : exam 1 trip.2

CP : exam 2 trip.2

total = 161

PP : 1 trip.2

SP : 8 trip.2

FP : 8 trip.2

O2 : 1 trip.2

CP : 2 trip.2

118 total

PP : exam trip.2

SP : 1 trip.2

FP : 1 trip.2

O2 : 1 trip.2

CP : 1 trip.2

LAB-07:-

- Q. WAP that demonstrates handling of exceptions in inheritance tree.
 Create a base class called "Father" and derived class called "son" which extends the base class. In Father class, implement a constructor which takes the age and throws the exception `WrongAge` when the input age < 0. In son class implement a constructor that takes both father and son's age and throws an exception if son's age is \geq father's age.

SOP

```
import java.util.Scanner;
class WrongAge extends Exception
{
    WrongAge (String string message)
    {
        Super (message);
    }
}
```

Class Father

```
int age;
Father (int age) throws WrongAge
{
    if (age < 0)
        throw new WrongAge ("Age cannot be negative");
}
```

```
if (age < 0)
    throw new WrongAge ("Age cannot be negative");
}
```

(Fo-9/1)

this. age = age; // make both same

{ } (if both are same then

both "no" both ends and no change)

else throw "no" both ends because

class Son extends Father and all
method from superclass is overridden

int Sage; int son age;

Son (int father Age, int son Age)

throws Wrong Age

Super (father Age);

y (son Age) = father Age

throw new Wrong Age;

Wrong Age ("Son's Age is not father's Should be
less than Father's age");

this. sage = son Age;

public class Age

{ public static void main (String args [])

Scanner sc = new Scanner (System. in);
try {

System.out.println ("Enter the Father's
age : ");

int b = sc. nextInt ();

Son son = new Son (a, b);

Catch (Wrong Age e)

```
System.out.println("Exception : "+  
    e.getMessage());
```

- Output -

» Enter the Father's age

58

Enter the son's age

37

Son's age is 37

» Enter the Father's age

27

Enter the Son's age

27

Father's age should be greater than
Son's age.

» Enter the Father's age

18

Enter the Son's age

35

Father's age should be greater than
Son's age.

✓
31.01.24

LAB-B:-

Q. Write a program which creates two threads, one thread displaying "BMS College of Engineering" once every ten seconds and another displaying ("CSE" Once every two seconds.

Sol:

Public class Thread Example

{

public static void main (String [] args)

{

Thread thread1 = new Thread (new

Display Every Ten seconds ());

Thread thread2 = new Thread (new

Display Every Two seconds ());

thread1.start();

thread2.start();

}

}

Class Display Every Ten seconds implements Runnable {

public void run()

{

while (true)

{

try {

Thread.sleep(1000);

System.out.println("BMS college of

Engineering");

}

```
catch (InterruptedException e)
```

```
{  
    e.printStackTrace();  
}
```

```
}
```

```
}
```

class Display Every Two Seconds implements Runnable

```
{
```

```
    public void run()
```

```
{  
    while (true)  
    {  
        try  
        {  
            Thread.sleep(2000);  
            System.out.println("CSE");  
        }  
        catch (InterruptedException e)  
        {  
            e.printStackTrace();  
        }  
    }  
}
```

Output:-

CSE

CSE

CSE

CSE

BMS college of Engineering

CSE

CSE

CSE

CSE

BMS college of Engineering

CSE

CSE

CSE

CSE

BMS college of Engineering.

LAB-10>

Q. Write a Program to Demonstrate Inter process communication and deadlock.

Sol:

Class Q

{

int n;

boolean valueSet = false;

Synchronized int get()

{ while (!valueSet)

try {

System.out.println("In Consumer
Waiting " + n);

Wait();

}

catch (InterruptedException e)

{

System.out.println("Interrupted
Exception Caught");

}

System.out.println("got:" + n);

valueSet = true;

System.out.println("In Intimate
Producer " + n);

notify();

return n;

}

Synchronized void put(int n)

{ while (valueSet)

```
try
{
    System.out.println ("In Producer writing");
    wait();
}
catch (InterruptedException e)
{
    System.out.println ("Interrupted Exception Caught");
}
this.n=n;
valueset=true;
System.out.println ("Put :" + n);
System.out.println ("In Intimate Consumer loop");
}
```

Class Producer implements Runnable

```
{}
Producer (Q q)
{
    this.q=q;
    new Thread (this, "Producer").start();
}
```

public void run()

{ int i=0;

while (i<15)

{ bin[i] = bin[i] + 1;
 q.put(i++);

}

3
3
3

Class consumer implements Runnable

{

Q q;

Consumer(Q q)

{

this.q = q;

new Thread(this, "consumer").start();

}

public void run()

{

int i=0;

while(i<15)

{

int n=q.get();

System.out.println("Item consumed: "+n);

i++;

}

}

}

class PCFired

{

Public static void main(String args[])

{

Q q=new Q();

new Producer(q);

new Consumer(q);

System.out.println("Press Control-c
to stop.");

}

∴ Output :-

Press Control -c to stop

Put: 0

Intimate consumer

Producer waiting

Get = 0

Intimate Producer

put : 1

Intimate consumer

Producer waiting

Consumed: 0

got: 1

Intimate Producer

(consumed): 1

Put: 2

★ Deadlock :-

```

>> class A {
    synchronized void foo(B b) {
        String name = Thread.currentThread().getName();
        System.out.println(name + " entered A.foo()");
        try {
            Thread.sleep(1000);
        } catch (Exception e) {
            System.out.println("A interrupted");
            System.out.println(name + " trying to call B.last()");
            b.last();
        }
        void last() {
            System.out.println("Inside A.last()");
        }
    }
}

class B {
    synchronized void bar(A a) {
        String name = Thread.currentThread().getName();
        System.out.println(name + " entered B.bar()");
        a.last();
    }
}
  
```

```
System.out.println("name + "entered B.bau");
try {
    Thread.sleep(1000);
}
catch (Exception e) {
    System.out.println("B interrupted");
}
System.out.println("name + "trying to call
    A.last()");
a.last();
}
```

```
void last() {
}
System.out.println("Inside A.last");
}
```

```
class Deadlock implements Runnable
```

```
A a = new A();
```

```
B b = new B();
```

```
Deadlock()
```

```
{
```

```
    Thread.currentThread().setName
        ("main Thread");
    Thread t = new Thread(this, "Racing"
        Thread t.start();
    t.start();
    a.foo(b);
}
```

```
System.out.println("Back in
    main thread");
```

}

Public void run ()
{

b. bar (a);

System.out.println ("Back in other
thread");

}

Public static void main (String args [])

{

new Dead Lock ();

}

}

- Output -

Main Thread entered A. foo

Racing Thread entered B. bar

main Thread trying to call B. last ()

Inside A. ~~last~~

Back in main Thread

Racing Thread trying to call A. last ()

Inside A. ~~last~~

Back in other thread.

St
13.02.24

//

20/02/2024

LAB 09

- Q Write a program that creates a user interface to perform integer divisions.

SOL

```
import javax.swing.*;  
import java.awt.*;  
import java.awt.event.*;
```

```
class SwingDemo
```

```
{  
    SwingDemo()  
    {
```

```
        JFrame jfrm = new JFrame("Divide  
APP");
```

```
        jfrm.setSize(275, 150);  
        jfrm.setLayout(new FlowLayout());  
        jfrm.setDefaultCloseOperation(JFrame.  
        EXIT_ON_CLOSE);
```

```
        JLabel jlab = new JLabel("Enter the  
divider and dividend:");
```

```
        JTextField afty = new JTextField(8);  
        JTextField btyp = new JTextField(8);
```

```
        JButton button = new JButton  
        ("Calculate");
```

```
        JLabel em = new JLabel();
```

```
        JLabel alab = new JLabel();
```

```
        JLabel blab = new JLabel();
```

```
        JLabel onslab = new JLabel();
```

```
jfm.add(cnf);
jfm.add(jlab);
jfm.add(cjtf);
jfm.add(bjtf);
jfm.add(button);
jfm.add(alab);
jfm.add(blab);
jfm.add(constab);
```

```
ActionListener l = new ActionListener()
{
```

```
    public void actionPerformed(ActionEvent evt)
```

```
    {
        System.out.println("Action event from  
a text field");
    }
},
```

```
cjtf.addActionListener(l);
bjtf.addActionListener(l);
```

```
button.addActionListener(new ActionListener())
{
```

```
    public void actionPerformed(ActionEvent evt)
    {
        System.out.println("Action event from  
a button");
    }
},
```

```
public void actionPerformed(ActionEvent evt) {
```

```
    try {
        int a = Integer.parseInt(cjtf.getText());
        int b = Integer.parseInt(bjtf.getText());
        int ans = a * b;
        System.out.println("Answer is " + ans);
    }
}
```

```
int a = Integer.parseInt(cjtf.getText());
int b = Integer.parseInt(bjtf.getText());
```

```
int ans = a * b;
```

```
alab.setText("A=" + a);
blab.setText("B=" + b);
anilab.setText("Am=" + am);
}
```

```
catch (NumberFormatException e)
```

```
{  
    alab.setText("E");
    blab.setText("E");
    anilab.setText("E");
    err.setText("Enter only Integer!");
}
```

```
catch (ArithmaticException e)
```

```
{  
    alab.setText("E");
    blab.setText("E");
    anilab.setText("E");
    err.setText("B should be NON  
zero!");
}
```

```
});
```

```
jfm.setVisible(true);
```

```
public static void main (String args[])
{
    SwingUtilities.invokeLater(new
        Runnable()
    {

```

```
        public void run()
    
```

```
        new swingDemo();
    }
});
```

```
}
```

~~Output :-~~

Enter the divisor and dividend

[48]

[6]

[calculate]

A=48 B=6 Ans=8 //

LAB PROGRAM 1

Develop a Java program that prints all real solutions to the quadratic equation $ax^2 + bx + c = 0$. Read in a, b, c and use the quadratic formula. If the discriminate $b^2 - 4ac$ is negative, display a message stating that there are no real solutions.

```
import java.util.Scanner;
class Quadratic
{
    int a, b, c;
    double r1, r2, d;
    void getd()
    {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter the coefficients of a,b,c");
        a = s.nextInt();
        b = s.nextInt();
        c = s.nextInt();
    }
    void compute()
    {
        while(a==0)
        {
            System.out.println("Not a quadratic equation");
            System.out.println("Enter a non zero value for a:");
            Scanner s = new Scanner(System.in);
            a = s.nextInt();
        }
        d = b*b-4*a*c;
        if(d==0)
```

```

{
r1 = (-b)/(2*a);
System.out.println("Roots are real and equal");
System.out.println("Root1 = Root2 = " + r1);
}
else if(d>0)
{
r1 = ((-b)+(Math.sqrt(d)))/(double)(2*a);
r2 = ((-b)-(Math.sqrt(d)))/(double)(2*a);
System.out.println("Roots are real and distinct");
System.out.println("Root1 = " + r1 + " Root2 = " + r2);
}
else if(d<0)
{
System.out.println("Roots are imaginary");
r1 = (-b)/(2*a);
r2 = Math.sqrt(-d)/(2*a);
System.out.println("Root1 = " + r1 + " + i" + r2);
System.out.println("Root1 = " + r1 + " - i" + r2);
}
}

class QuadraticMain
{
public static void main(String args[])
{
Quadratic q = new Quadratic();
q.getd();
q.compute();
}
}

```

LAB PROGRAM 2

Develop a Java program to create a class Student with members usn, name, an array credits and an array marks. Include methods to accept and display details and a method to calculate SGPA of a student.

```
import java.util.Scanner;
class subject{
    int subjectMarks, credits, grade;
}
class Student {
    String name;
    String usn;
    double SGPA;
    Scanner s;
    subject subjects[];
}

Student()
{
    int i;
    subjects = new subject[9];
    for(i=0;i<8;i++)
        subjects[i] = new subject();
    s = new Scanner(System.in);
}

public void getStudentDetails(){
    System.out.println("Enter student name:");
    name=s.nextLine();
    System.out.println("Enter Student USN:");
    usn=s.nextLine();
}

public void getMarks(){
    int i;
    for(i=0;i<8;i++){

```

```

System.out.println("Enter marks of subject"+(i+1)+":");
subjects[i].subjectMarks= s.nextInt();
if(subjects[i].subjectMarks>=40&&subjects[i].subjectMarks<=100){
subjects[i].grade=calculateGrade(subjects[i].subjectMarks);}
else{
System.out.println("Invalid Marks. Marks should be between 40 and 100");}
System.out.println("enter credits:");
subjects[i].credits=s.nextInt();
}
}

public int calculateGrade(int marks){
if (marks>=90)
return 10;
else if(marks>=70&&marks<=80)
return 9;
else if(marks>=60&&marks<=70)
return 8;
else if(marks>=50&&marks<=60)
return 7;
else
return 6;
}

public void computeSGPA() {
int totalscore = 0;
int totalcred = 0;
for (int i = 0; i < 8; i++) {
totalscore += subjects[i].grade * subjects[i].credits;
totalcred += subjects[i].credits;
}
SGPA = (double) totalscore / (double) totalcred;
}

```

```
}

class Stud{
    public static void main(String args[]){
        Student s1=new Student();
        s1.getStudentDetails();
        s1.getMarks();
        s1.computeSGPA();
        System.out.println("Student name:"+s1.name);
        System.out.println("Student usn:"+s1.usn);

        System.out.println("Student sgpa:"+s1.SGPA);}
    }
```

LAB PROGRAM 3

Create a class Book which contains four members: name, author, price, num_pages. Include a constructor to set the values for the members. Include methods to set and get the details of the objects. Include a toString() method that could display the complete details of the book. Develop a Java program to create n book objects.

```
import java.util.Scanner;

class Book {

    private String name;
    private String author;
    private double price;

    private int numPages;

    public Book(String name, String author, double price, int numPages) {
        this.name = name;
        this.author = author;
        this.price = price;
        this.numPages = numPages;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public String getAuthor() {
        return author;
    }

    public void setPrice(double price) {
```

```

        this.price = price;
    }

    public double getPrice() {
        return price;
    }

    public void setNumPages(int numPages) {
        this.numPages = numPages;
    }

    public int getNumPages() {
        return numPages;
    }

    public String toString() {
        return "Book Details: \nName: " + name + "\nAuthor: " + author + "\nPrice: INR" + price +
        "\nNumber of Pages: " + numPages;
    }
}

public class Main {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the number of books: ");
        int n = scanner.nextInt();
        Book[] books = new Book[n];
        for (int i = 0; i < n; i++) {
            System.out.println("\nEnter details for Book " + (i + 1) + ":");
            scanner.nextLine();
            System.out.println("Enter name: ");
            String name = scanner.nextLine();
            System.out.println("Enter author: ");
            String author = scanner.nextLine();
            System.out.println("Enter price: ");
            double price = scanner.nextDouble();
    
```

```
System.out.println("Enter number of pages: ");
int numPages = scanner.nextInt();
books[i] = new Book(name, author, price, numPages);
}

System.out.println("\nDetails of all books:");
for (int i = 0; i < n; i++) {
    System.out.println("\nBook " + (i + 1) + ":\n" + books[i]);
}
scanner.close();
}

}
```

LAB PROGRAM 4

Develop a Java program to create an abstract class named Shape that contains two integers and an empty method named printArea(). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the classShape. Each one of the classes contain only the method printArea() that prints the area of the given shape.

```
import java.util.Scanner;
class InputScanner {
    Scanner s = new Scanner(System.in);
    int getInput(String prompt) {
        System.out.println(prompt);
        return s.nextInt();
    }
}
class shape extends InputScanner {
    double dim1;
    double dim2;
    shape(double a, double b) {
        dim1 = a;
        dim2 = b;
    }
}
class Rectangle extends shape {
    Rectangle() {
        super(0, 0);
        dim1 = getInput("Enter length");
        dim2 = getInput("Enter breadth");
    }
    double area() {
        System.out.println("Inside Area for Rectangle.");
        return dim1 * dim2;
    }
}
```

```

    }

}

class Triangle extends shape {

    Triangle() {
        super(0, 0);
        dim1 = getInput("Enter length");
        dim2 = getInput("Enter base");
    }

    double area() {
        System.out.println("Inside Area for Triangle.");
        return dim1 * dim2 / 2;
    }
}

class Circle extends shape {

    Circle() {
        super(0, 0);
        dim1 = getInput("Enter the radius");
        dim2 = dim1;
    }

    double area() {
        System.out.println("Inside Area for Circle.");
        return Math.PI * dim1 * dim2;
    }
}

public class Areas {
    public static void main(String[] args) {
        Rectangle rectangle = new Rectangle();
        System.out.println("Area of Rectangle: " + rectangle.area());

        Triangle triangle = new Triangle();
        System.out.println("Area of Triangle: " + triangle.area());
    }
}

```

```
Circle circle = new Circle();
System.out.println("Area of Circle: " + circle.area());
}
}
```

LAB PROGRAM 5

Develop a Java program to create a class Bank that maintains two kinds of account for its customers, one called savings account and the other current account. The savings account provides compound interest and withdrawal facilities but no cheque book facility. The current account provides cheque book facility but no interest. Current account holders should also maintain a minimum balance and if the balance falls below this level, a service charge is imposed. Create a class Account that stores customer name, account number and type of account. From this derive the classes Cur-acct and Sav-acct to make them more specific to their requirements. Include the necessary methods in order to achieve the following tasks:

- a) Accept deposit from customer and update the balance.**
- b) Display the balance.**
- c) Compute and deposit interest**
- d) Permit withdrawal and update the balance**

Check for the minimum balance, impose penalty if necessary and update the balance.

```
import java.util.Scanner;
class Account {
    String customerName;
    int accountNumber;
    String accountType;
    double balance;
    Account(String name, int number, String type, double initialBalance) {
        customerName = name;
        accountNumber = number;
        accountType = type;
        balance = initialBalance;
    }
    void deposit(double amount) {
        balance += amount;
        System.out.println("Deposit of INR " + amount + " successful");
    }
}
```

```

}

void displayBalance() {
    System.out.println("Account Number: " + accountNumber);
    System.out.println("Customer Name: " + customerName);
    System.out.println("Account Type: " + accountType);
    System.out.println("Balance: INR " + balance);
}

void withdraw(double amount) {
    if (balance >= amount) {
        balance -= amount;
        System.out.println("Withdrawal of INR " + amount + " successful");
    } else {
        System.out.println("Insufficient funds");
    }
}

void computeInterest() {
}

void checkMinimumBalance(double minBalance, double serviceCharge) {
}

class SavAcct extends Account {
    double interestRate = 0.05;

    SavAcct(String name, int number, String type, double initialBalance) {
        super(name, number, type, initialBalance);
    }

    void computeInterest() {
        double interest = balance * interestRate;
        balance += interest;
        System.out.println("Interest of INR " + interest + " added to the account");
    }
}

```

```

}

class CurAcct extends Account {
    double minBalance = 1000;
    double serviceCharge = 50;
    CurAcct(String name, int number, String type, double initialBalance) {
        super(name, number, type, initialBalance);
    }
    void checkMinimumBalance(double minBalance, double serviceCharge) {
        if (balance < minBalance) {
            System.out.println("Service charge of INR " + serviceCharge + " imposed");
            balance -= serviceCharge;
        }
    }
}
public class Bank {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of users: ");
        int numUsers = scanner.nextInt();
        Account[] accounts = new Account[numUsers];
        for (int i = 0; i < numUsers; i++) {
            System.out.println("\nUser " + (i + 1));
            System.out.print("Enter customer name: ");
            scanner.nextLine();
            String name = scanner.nextLine();
            System.out.print("Enter account number: ");
            int accNumber = scanner.nextInt();
            System.out.print("Enter initial deposit amount: INR ");
            double initialDeposit = scanner.nextDouble();
            System.out.print("Enter account type (Savings/Current): ");
            scanner.nextLine();
        }
    }
}

```

```

String accType = scanner.nextLine();
if (accType.equalsIgnoreCase("Savings")) {
    accounts[i] = new SavAcct(name, accNumber, accType, initialDeposit);
} else if (accType.equalsIgnoreCase("Current")) {
    accounts[i] = new CurAcct(name, accNumber, accType, initialDeposit);
} else {
    System.out.println("Invalid account type entered. Defaulting to Account.");
    accounts[i] = new Account(name, accNumber, "Account", initialDeposit);
}
}

boolean exit = false;
while (!exit) {
    System.out.println("\nChoose an option:");
    System.out.println("1. Deposit");

    System.out.println("2. Withdraw");
    System.out.println("3. Display Balance");
    System.out.println("4. Compute Interest (Savings only)");
    System.out.println("5. Exit");

    System.out.print("Enter your choice: ");
    int choice = scanner.nextInt();
    switch (choice) {
        case 1:
            System.out.print("Enter account number: ");
            int accNum = scanner.nextInt();
            System.out.print("Enter deposit amount: INR ");
            double depositAmount = scanner.nextDouble();
            for (Account acc : accounts) {
                if (acc.accountNumber == accNum) {
                    acc.deposit(depositAmount);
                }
            }
    }
}

```

```

break;

case 2:
    System.out.print("Enter account number: ");
    accNum = scanner.nextInt();
    System.out.print("Enter withdrawal amount: INR ");
    double withdrawAmount = scanner.nextDouble();
    for (Account acc : accounts) {
        if (acc.accountNumber == accNum) {
            acc.withdraw(withdrawAmount);
        }
    }
    break;

case 3:
    System.out.print("Enter account number: ");
    accNum = scanner.nextInt();
    for (Account acc : accounts) {
        if (acc.accountNumber == accNum) {
            acc.displayBalance();
        }
    }
    break;

case 4:
    System.out.print("Enter account number (for Savings account): ");
    accNum = scanner.nextInt();
    for (Account acc : accounts) {
        if (acc.accountNumber == accNum && acc instanceof SavAcct) {
            ((SavAcct) acc).computeInterest();
        }
    }
    break;

case 5:

```

```
    exit = true;
    break;
default:
    System.out.println("Invalid choice. Please enter a valid option.");
}
}
}
}
```

LAB PROGRAM 6

Create a package CIE which has two classes- Student and Internals. The class Personal has members like usn, name, sem. The class Internals has an array that stores the internal marks scored in five courses of the current semester of the student. Create another package SEE which has the class External which is a derived class of Student. This class has an array that stores the SEE marks scored in five courses of the current semester of the student. Import the two packages in a file that declares the final marks of n students in all five courses.

```
package CIE;  
public class Student {  
    public String usn;  
    public String name;  
    public int sem;  
    public Student(String usn, String name, int sem) {  
        this.usn = usn;  
        this.name = name;  
        this.sem = sem;  
    }  
}  
  
package CIE;  
public class Internals extends Student {  
    public int[] internalMarks;  
    public Internals(String usn, String name, int sem, int[] internalMarks) {  
        super(usn, name, sem);  
        this.internalMarks = internalMarks;  
    }  
}  
  
package SEE;
```

```

import CIE.Student;
public class External extends Student {
    public int[] seeMarks;
    public External(String usn, String name, int sem, int[] seeMarks) {
        super(usn, name, sem);
        this.seeMarks = seeMarks;
    }
}

```

```

import CIE.Internals;
import SEE.External;
import java.util.Scanner;
public class FinalMarks {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of students: ");
        int n = scanner.nextInt();
        Internals[] cieStudents = new Internals[n];
        External[] seeStudents = new External[n];
        for (int i = 0; i < n; i++) {
            System.out.println("Enter details for CIE of student " + (i + 1));
            System.out.print("USN: ");
            String usn = scanner.next();
            System.out.print("Name: ");
            String name = scanner.next();
            System.out.print("Semester: ");
            int sem = scanner.nextInt();
            int[] cieMarks = new int[5];
            System.out.print("Enter CIE marks for 5 courses: ");
            for (int j = 0; j < 5; j++) {

```

```

        cieMarks[j] = scanner.nextInt();
    }

    cieStudents[i] = new Internals(usn, name, sem, cieMarks);
}

for (int i = 0; i < n; i++) {
    System.out.println("Enter details for SEE of student " + (i + 1));
    System.out.print("USN: ");
    String usn = scanner.next();
    System.out.print("Name: ");
    String name = scanner.next();
    System.out.print("Semester: ");
    int sem = scanner.nextInt();
    int[] seeMarks = new int[5];
    System.out.print("Enter SEE marks for 5 courses: ");
    for (int j = 0; j < 5; j++) {
        seeMarks[j] = scanner.nextInt();
    }
    seeStudents[i] = new External(usn, name, sem, seeMarks);
}

System.out.println("\nFinal Marks of Students:");
for (int i = 0; i < n; i++) {
    System.out.println("\nDetails of Student " + (i + 1));
    System.out.println("USN: " + cieStudents[i].usn);
    System.out.println("Name: " + cieStudents[i].name);
    System.out.println("Semester: " + cieStudents[i].sem);
    System.out.println("CIE Marks: ");
    for (int j = 0; j < 5; j++) {
        System.out.print(cieStudents[i].internalMarks[j] + " ");
    }
    System.out.println("\nSEE Marks: ");
}

```

```
for (int j = 0; j < 5; j++) {  
    System.out.print(seeStudents[i].seeMarks[j] + " ");  
}  
}  
}  
}
```

LAB PROGRAM 7

Write a program that demonstrates handling of exceptions in inheritance tree. Create a base class called “Father” and derived class called “Son” which extends the base class. In Father class, implement a constructor which takes the age and throws the exception WrongAge() when the input age<0. In Son class, implement a constructor that cases both father and son’s age and throws an exception if son’s age is >=father’s age.

```
import java.util.Scanner;
class WrongAge extends Exception {
    public WrongAge(String message) {
        super(message);
    }
}
class Father {
    protected int fatherAge;
    public Father(int age) throws WrongAge {
        fatherAge = age;
        if (fatherAge < 0) {
            throw new WrongAge("Father's age cannot be negative");
        }
    }
}
class Son extends Father {
    private int sonAge;
    public Son(int fatherAge, int sonAge) throws WrongAge {
        super(fatherAge);
        this.sonAge = sonAge;
        if (sonAge <= 0) {
            throw new WrongAge("Son's age cannot be negative or zero");
        }
        if (sonAge >= fatherAge) {
```

```

        throw new WrongAge("Son's age cannot be greater than or equal to father's age");
    }
}
}

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        try {
            System.out.print("Enter father's age: ");
            int fatherAge = scanner.nextInt();
            System.out.print("Enter son's age: ");
            int sonAge = scanner.nextInt();
            Son son = new Son(fatherAge, sonAge);
            System.out.println("Father's age: " + fatherAge);
            System.out.println("Son's age: " + sonAge);
        } catch (WrongAge e) {
            System.out.println("Exception caught: " + e);
            System.out.println("Exception caught: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Error: " + e);
            System.out.println("Error: " + e.getMessage());
        } finally {
            scanner.close();
        }
    }
}

```

LAB PROGRAM 8

Write a program which creates two threads, one thread displaying “BMS College of Engineering” once every ten seconds and another displaying “CSE” once every two seconds.

```
class DisplayThread extends Thread {  
    private String message;  
    private int interval;  
    private boolean running = true;  
    public DisplayThread(String message, int interval) {  
        this.message = message;  
        this.interval = interval;  
    }  
    public void run() {  
        while (running) {  
            System.out.println(message);  
            try {  
                Thread.sleep(interval);  
            } catch (InterruptedException e) {  
                e.printStackTrace();  
            }  
        }  
    }  
    public void stopThread() {  
        running = false;  
    }  
}  
public class ThreadEx {  
    public static void main(String[] args) {  
        DisplayThread bmsThread = new DisplayThread("BMS College of Engineering", 10000);  
        DisplayThread cseThread = new DisplayThread("CSE", 2000);  
    }  
}
```

```
bmsThread.start();
cseThread.start();
System.out.println("Press Enter to stop the threads...");
try {
    System.in.read();
} catch (Exception e) {
    e.printStackTrace();
}
bmsThread.stopThread();
cseThread.stopThread();
}
```

LAB PROGRAM 9

Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields, Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw a NumberFormatException. If Num2 were Zero, the program would throw an ArithmeticException. Display the exception in a message dialog box.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
class SwingDemo{
SwingDemo(){
JFrame jfrm = new JFrame("Divider App");
jfrm.setSize(275, 150);
jfrm.setLayout(new FlowLayout());
jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
JLabel jlab = new JLabel("Enter the divider and divident:");
JTextField ajtf = new JTextField(8);
JTextField bjtf = new JTextField(8);
JButton button = new JButton("Calculate");
JLabel err = new JLabel();
JLabel alab = new JLabel();
JLabel blab = new JLabel();
JLabel anslab = new JLabel();
jfrm.add(err); // to display error bois
jfrm.add(jlab);
jfrm.add(ajtf);
jfrm.add(bjtf);
jfrm.add(button);
```

```

jfrm.add(alab);
jfrm.add(blab);
jfrm.add(anslab);

ActionListener l = new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        System.out.println("Action event from a text field");
    }
};

ajtf.addActionListener(l);
bjtf.addActionListener(l);
button.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        try{
            int a = Integer.parseInt(ajtf.getText());
            int b = Integer.parseInt(bjtf.getText());
            int ans = a/b;

            alab.setText("\nA = " + a);
            blab.setText("\nB = " + b);
            anslab.setText("\nAns = " + ans);
        }
        catch(NumberFormatException e){
            alab.setText("");
            blab.setText("");
            anslab.setText("");
            err.setText("Enter Only Integers!");
        }
        catch(ArithmeticException e){
            alab.setText("");
            blab.setText("");
            anslab.setText("");
            err.setText("B should be NON zero!");
        }
    }
}

```

```
}

}

});

jfrm.setVisible(true);

}

public static void main(String args[]){

SwingUtilities.invokeLater(new Runnable(){

public void run(){

new SwingDemo();

}

});

}

}
```

LAB PROGRAM 10

Demonstrate Inter process Communication and deadlock.

IPC

```
class Q {  
    int n;  
    boolean valueSet = false;  
    synchronized int get() {  
        while(!valueSet)  
            try {  
                wait();  
            } catch(InterruptedException e) {  
                System.out.println("InterruptedException caught");  
            }  
        System.out.println("Got: " + n);  
        valueSet = false;  
        notify();  
        return n;  
    }  
    synchronized void put(int n) {  
        while(valueSet)  
            try {  
                wait();  
            } catch(InterruptedException e) {  
                System.out.println("InterruptedException caught");  
            }  
        this.n = n;  
        valueSet = true;  
        System.out.println("Put: " + n);  
        notify();  
    }  
}
```

```

}

}

class Producer implements Runnable {
    Q q;
    Producer(Q q) {
        this.q = q;
        new Thread(this, "Producer").start();
    }
    public void run() {
        int i = 0;
        while(i<15) {
            q.put(i++);
        }
    }
}

class Consumer implements Runnable {
    Q q;
    Consumer(Q q) {
        this.q = q;
        new Thread(this, "Consumer").start();
    }
    public void run() {
        int i=0;
        while(i<15) {
            int r=q.get();
            i++;
        }
    }
}

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

```

```
Q q = new Q();  
new Producer(q);  
new Consumer(q);  
System.out.println("Press Control-C to stop.");  
}  
}
```

Deadlock

```
class A {  
    synchronized void foo(B b) {  
        String name = Thread.currentThread().getName();  
        System.out.println(name + " entered A.foo");  
        try {  
            Thread.sleep(1000);  
        } catch(Exception e) {  
            System.out.println("A Interrupted");  
        }  
        System.out.println(name + " trying to call B.last()");  
        b.last();  
    }  
    void last() {  
        System.out.println("Inside A.last");  
    }  
}  
  
class B {  
    synchronized void bar(A a) {  
        String name = Thread.currentThread().getName();  
        System.out.println(name + " entered B.bar");  
        try {  
            Thread.sleep(1000);  
        } catch(Exception e) {  
            System.out.println("B Interrupted");  
        }  
        System.out.println(name + " trying to call A.last()");  
        a.last();  
    }  
    void last() {  
}
```

```
System.out.println("Inside A.last");
}
}

class Deadlock implements Runnable
{
A a = new A();
B b = new B();

Deadlock() {
    Thread.currentThread().setName("MainThread");
    Thread t = new Thread(this,"RacingThread");
    t.start();
    a.foo(b);
    System.out.println("Back in mainthread");
}

public void run() {
    b.bar(a);
    System.out.println("Back in other thread");
}

public static void main(String args[]) {
    new Deadlock();
}
}
```