1.CarSale

```
import java.util.Scanner;
class CarSale {

    int sales[][] =new int [6][12];
    //memmber of function
    //read sales data
    Scanner s=new Scanner (System.in);
    void read_sales(){
        System.out.println("enter the sales data for each car : ");
        for(int i=0;i<6;i++)
        {
            for(int j=0;j<12;j++)
            {
               sales[i][j]=s.nextInt();
            }
        }
    }

    void display_Sale()
{
    System.out.println("Sales data for car manufacturers:");
    for(int i=0;i<6;i++)
    {
        for(int j=0;j<12;j++)
        {
            System.out.print(sales[i][j] + "\t"); // Use print instead of
println for tab separation
        }
        System.out.println(); // Move to the next line after each manufacturer
    }
}

    void avg_sales(){
        int total;
        for(int i=0;i<6;i++){
            total=0;
            for(int j=0;j<12;j++){
                total+=sales[i][j];
            }
            System.out.println("Average sale for manufacturer " + i + " is: "
+ (total / 12.0)); // Use 12.0 for floating-point division
        }
    }
     void total_sales(){
        System.out.println();
        int total;
        for(int i=0;i<6;i++){
```

```
            total=0;
            for(int j=0;j<12;j++){
                total+=sales[i][j];
            }
            System.out.println("Total sales for manufacturer " + i +" is:
"+total);
        }
     }

    void month_sales(){
        System.out.println();
        int msales;
        for(int i=0;i<12;i++){
            msales=0;
            for(int j=0;j<6;j++){
                msales=msales+sales[j][i];
            }
            System.out.println("month sales for manufacturer " +i+ "is :
"+msales);
        }
    }

        void max_sales(){
            int mfgr,max=-1,month =0;
            System.out.println("enter car mfgr");
            mfgr =s.nextInt();
            for(int i=0;i<12;i++)
            {
                if(sales[mfgr][i]>max){
                    max=sales[mfgr][i];
                    month=i+1;
                }
            }
            System.out.println(" month is :" +month);
    }
}
public class CarSaleDemo{
    public static void main(String args[]){
        CarSale MS = new CarSale();
        MS.read_sales();
        MS.display_Sale();
        MS.avg_sales();
        MS.total_sales();
        MS.month_sales();
        MS.max_sales();
    }
}
```

Triangle

```
import java.util.Scanner;

class myTriangle{
        //data members
        double S1,S2,S3,area;
        void init(){
            Scanner s=new Scanner(System.in);
            System.out.println("Enter the side of the Triangle");
            S1=s.nextDouble();
            S2=s.nextDouble();
            S3=s.nextDouble();
        }


        void type(){
            if((S1==S2)&&(S2==S3))
                System.out.println("Equilateral Triangle");
            else if((S1==S2)||(S2==S3)||(S3==S1))
                System.out.println("Isosceles Triangle");
            else
                System.out.println("Scalene Triangle");
        }

        void FindArea(){
            double S=(S1+S2+S3)/2;
            if(((S1+S2)<S3)||((S2+S3)<S1)||((S3+S1)<S2))
            {
                System.out.println("ERROR!!   Triangle cannot be formed");
            }
            else{
                area=Math.sqrt(S*(S-S1)*(S-S2)*(S-S3));
                display();
            }
        }

        void display(){
            System.out.println("Area of the triangle is:"+area);
        }
}
public class TriangleDemo {
    public static void main(String args[]){
    myTriangle T1=new myTriangle();//creating a object
        T1.init();
        T1.type();
        T1.FindArea();
    }
}
```

Book

```
class Book {
    String Title;
    String Author;
    String Publisher;
    double Price;
    double Discount;
    double Dprice;

    Book(){
    }
    Book(String T, String A, String P, double price) {
        Title = T;
        Author = A;
        Publisher = P;
        Price = price;
    }
    void discountPrice() {
        Discount = Price * 0.10;
        Dprice=Price-Discount;
    }

    void display() {
        System.out.println("Title: " + Title);
        System.out.println("Author: " + Author);
        System.out.println("Publisher: " + Publisher);
        System.out.println("Price: " + Price);
        System.out.println("Discount: " + Discount);
        System.out.println("Discounted Price: " + Discount);
        System.out.println();

    }
}

class KidsStory extends Book {
    KidsStory(String T, String A, String P, double price) {
        Title = T;
        Author = A;
        Publisher = P;
        Price = price;
    }
    @Override
    void discountPrice() {
        Discount = Price * 0.10;
        Dprice=Price-Discount;
    }
}
```

```
class Gk extends Book {
    Gk(String T, String A, String P, double price) {
        Title = T;
        Author = A;
        Publisher = P;
        Price = price;
    }
    @Override
    void discountPrice() {
        Discount = Price * 0.20;
        Dprice=Price-Discount;
    }
}

class Scientific extends Book {
    Scientific(String T, String A, String P, double price) {
        Title = T;
        Author = A;
        Publisher = P;
        Price = price;
    }
    @Override
    void discountPrice() {
       Discount = Price * 0.25;
       Dprice=Price-Discount;
    }
}

class BookDemo {
    public static void main(String[] args) {
        Book t1= new Book("Book", "Name", "PName", 90.0);
        KidsStory ks1 = new KidsStory("Harry", "Abc", "Xyz", 39.99);
        Gk gk1 = new Gk("GK Book", "Author Name", "Publisher Name", 20.0);
        Scientific sci1 = new Scientific("Science Book", "Scientist",
"ScienceHub", 55.0);

        ks1.discountPrice();
        gk1.discountPrice();
        sci1.discountPrice();

        ks1.display();
        gk1.display();
        sci1.display();
    }
}
```

Lsearch

```java
import java.util.Scanner;
class Lsearch {
    boolean Lsearch(int arr[], int ele) {
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] == ele) {
                System.out.println(ele + " is found at index " + i);
                return true;
            }
        }
        return false;
    }

    boolean Lsearch(double arr2[], double ele) {
        for (int i = 0; i < arr2.length; i++) {
            if (arr2[i] == ele) {
                System.out.println(ele + " is found at index " + i);
                return true;
            }
        }
        return false;
    }
    boolean Lsearch(char arr3[], char ele) {
        for (int i = 0; i < arr3.length; i++) {
            if (arr3[i] == ele) {
                System.out.println(ele + " is found at index " + i);
                return true;
            }
        }
        return false;
    }
}
class LinearSearchDemo {
    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5};
        double[] arr2 = {1.6, 2.8, 3.0, 4.3, 5.6};
        char[] arr3 = {'a', 'b', 'c', 'd', 'e'};
        Scanner s=new Scanner(System.in);
        System.out.println("enter the elements to search:");
        int key1=s.nextInt();
        double key2=s.nextDouble();
        char key3=s.next().charAt(0);
        Lsearch search = new Lsearch();
        search.Lsearch(arr, key1);
        search.Lsearch(arr2,key2);
        search.Lsearch(arr3,key3);
    }
}
```

Complex

```
class complex{
    double real,imag;

    complex(){
        real=imag=0;
    }
    complex(double R,double I){
        real=R;
        imag=I;
    }
    complex add(complex c1,complex c2){
        complex res=new complex();
        res.real=c1.real+c2.real;
        res.imag=c1.imag+c2.imag;
        return(res);
    }
    complex add(complex c1,int x){
        complex res=new complex();
        res.real=c1.real+x;
        res.imag=c1.imag;
        return (res);
    }

    void Display(){
        if(imag<0)
            System.out.println("Resultant complex numbers:"+real+"- i"+imag);
        else{
            System.out.println("Resultant complex numbers:"+real+"+ i"+imag);
        }
    }
}
public class ComplexDemo {
    public static void main(String[] args) {
        complex c1=new complex(1,8);
        complex c2=new complex(5,-4);

        c1.Display();
        c2.Display();

        complex result = c1.add(c1, c2);
        result.Display();

        complex result2 = c1.add(c2, -3);
        result2.Display();
    }
}
```

Queue

```
package queuedemo2;

interface Queue {
    void enqueue(int element);
    int dequeue();
    int peek();
    boolean isFull();
    boolean isEmpty();
    void display();
}

class LinearQueue implements Queue {
    private int[] arr;
    private int front;
    private int rear;
    private int size;

    LinearQueue(int size) {
        this.size = size;
        arr = new int[size];
        front = -1;
        rear = -1;
        System.out.println("The size of Linear Queue is: " + size);
    }

    @Override
    public void enqueue(int element) {
        if (isFull()) {
            System.out.println("Linear Queue is full!");
            return;
        }
        if (front == -1)
            front = 0;
        rear++;
        arr[rear] = element;
    }

    @Override
    public int dequeue() {
        if (isEmpty()) {
            System.out.println("Linear Queue is empty!");
            return -1;
        }
        int element = arr[front];
        front++;
        if (front > rear) {
            front = rear = -1;
```

```
        }
        return element;
    }

    @Override
    public int peek() {
        if (isEmpty()) {
            System.out.println("Linear Queue is empty!");
            return -1;
        }
        return arr[front];
    }

    @Override
    public boolean isFull() {
        return rear == size - 1;
    }

    @Override
    public boolean isEmpty() {
        return front == -1 || front > rear;
    }

    @Override
    public void display() {
        if (isEmpty()) {
            System.out.println("Linear Queue is empty.");
            return;
        }
        System.out.print("Linear Queue Elements are: ");
        for (int i = front; i <= rear; i++) {
            System.out.print(arr[i] + " ");
        }
        System.out.println();
    }
}

class CircularQueue implements Queue {
    private int[] arr;
    private int front;
    private int rear;
    private int size;

    CircularQueue(int size) {
        this.size = size;
        arr = new int[size];
        front = -1;
        rear = -1;
```

```
        System.out.println("The size of Circular Queue is: " + size);
    }

    @Override
    public void enqueue(int element) {
        if (isFull()) {
            System.out.println("Circular Queue is full!");
            return;
        }
        if (front == -1) {
            front = 0;
        }
        rear = (rear + 1) % size;
        arr[rear] = element;
    }

    @Override
    public int dequeue() {
        if (isEmpty()) {
            System.out.println("Circular Queue is empty!");
            return -1;
        }
        int element = arr[front];
        if (front == rear) {
            front = rear = -1;
        } else {
            front = (front + 1) % size;
        }
        return element;
    }

    @Override
    public int peek() {
        if (isEmpty()) {
            System.out.println("Circular Queue is empty!");
            return -1;
        }
        return arr[front];
    }

    @Override
    public boolean isFull() {
        return (rear + 1) % size == front;
    }

    @Override
    public boolean isEmpty() {
        return front == -1;
```

```
    }

    @Override
    public void display() {
        if (isEmpty()) {
            System.out.println("Circular Queue is empty.");
            return;
        }
        System.out.print("Circular Queue Elements are: ");
        int i = front;
        while (true) {
            System.out.print(arr[i] + " ");
            if (i == rear) break;
            i = (i + 1) % size;
        }
        System.out.println();
    }
}

class QueueDemo2 {
    public static void main(String[] args) {

        Queue linearQueue = new LinearQueue(3);
        linearQueue.enqueue(1);
        linearQueue.enqueue(2);
        linearQueue.enqueue(3);

        linearQueue.display();

        System.out.println("Front element of the Linear Queue is: " +
linearQueue.peek());

        System.out.println("The front element which is Dequeued: " +
linearQueue.dequeue());

        linearQueue.display();

        System.out.println("Peek element of Linear Queue is: " +
linearQueue.peek());




        Queue circularQueue = new CircularQueue(3);
        circularQueue.enqueue(7);
        circularQueue.enqueue(8);
        circularQueue.enqueue(9);
        circularQueue.enqueue(19);
```

```
        circularQueue.display();

        System.out.println("Front element of the Circular Queue is: " +
circularQueue.peek());

        System.out.println("The front element which is Dequeued: " +
circularQueue.dequeue());

        circularQueue.display();

        System.out.println("Peek element of Circular Queue is: " +
circularQueue.peek());


        circularQueue.display();



    }
}
```

License

```
package termwork4b;
import java.util.Scanner;
class AgeException extends Exception{
    AgeException(String msg){
        super(msg);
    }
}
class learnerLiscensException extends Exception{
    learnerLiscensException(String msg){
        super(msg);
    }
}
class AcceidentException extends Exception{
    AcceidentException(String msg){
        super(msg);
    }
}

public class Termwork4B {
    static void processApplication(int age,boolean VLL,boolean NOA) throws
 AgeException,learnerLiscensException,AcceidentException
    {
        if(age<18)
```

```
           throw new AgeException ("invalid age: user must be above 18 years
");
         if(!VLL)
              throw new learnerLiscensException ("invalid learner's licens  :
user must have valid lernaers licens ");
         if(!NOA)
          throw new  AcceidentException  ("invalid condition : user must have
no accident");
    }
     public static void main(String[] args) {
        // TODO code application logic here
        String name;
        boolean no_of_accd,valid_LL;
        int age;
        Scanner s=new Scanner(System.in);

         try{

        System.out.println("Enter user name");
         name=s.nextLine();

         System.out.println("Enter user age");
         age =s.nextInt();

         System.out.println("Does user have valid learner
liscens(true/false):");
         valid_LL =s.nextBoolean();

         System.out.println("Does user makes acceident(true/false):");
         no_of_accd =s.nextBoolean();

         processApplication(age,valid_LL,no_of_accd);
         }
         catch(AgeException |learnerLiscensException |AcceidentException ae){

             System.out.println("User cannot be Issued Liscens");
             System.out.println(ae.getMessage());

         }
         catch(Exception e){
             System.out.println("other Eception are:");
         }
    }//end of main


}
```

HashMap

```
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

public class HashMapMenu {
    public static void main(String[] args) {
        HashMap<String, String> loginCredentials = new HashMap<>();
        Scanner scanner = new Scanner(System.in);
        int choice;

        do {
            System.out.println("\n--- HashMap Menu ---");
            System.out.println("1. Add a key-value pair");
            System.out.println("2. Retrieve the value for a given key");
            System.out.println("3. Retrieve all keys");
            System.out.println("4. Retrieve all values");
            System.out.println("5. Retrieve all key-value pairs");
            System.out.println("6. Change the value associated with a key");
            System.out.println("7. Remove a HashMap element by key");
            System.out.println("8. Remove a HashMap entry by key and value");
            System.out.println("9. Check if a value exists in the HashMap");
            System.out.println("10. Display the entire HashMap");
            System.out.println("11. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();
            scanner.nextLine(); // Consume newline

            switch (choice) {
                case 1:
                    System.out.print("Enter key (username): ");
                    String key = scanner.nextLine();
                    if (loginCredentials.containsKey(key)) {
                        System.out.println("Error: Key already exists.");
                    } else {
                        System.out.print("Enter value (password): ");
                        String value = scanner.nextLine();
                        loginCredentials.put(key, value);
                        System.out.println("Key-value pair added
successfully.");
                    }
                    break;

                case 2:
                    System.out.print("Enter key (username): ");
                    key = scanner.nextLine();
                    if (loginCredentials.containsKey(key)) {
```

```
                        System.out.println("Value (password): " +
loginCredentials.get(key));
                    } else {
                        System.out.println("Error: Key not found.");
                    }
                    break;

                case 3:
                    System.out.println("All Keys: " +
loginCredentials.keySet());
                    break;

                case 4:
                    System.out.println("All Values: " +
loginCredentials.values());
                    break;

                case 5:
                    System.out.println("All Key-Value Pairs: " +
loginCredentials.entrySet());
                    break;

                case 6:
                    System.out.print("Enter key (username): ");
                    key = scanner.nextLine();
                    if (loginCredentials.containsKey(key)) {
                        System.out.print("Enter new value (password): ");
                        String newValue = scanner.nextLine();
                        loginCredentials.put(key, newValue);
                        System.out.println("Value updated successfully.");
                    } else {
                        System.out.println("Error: Key not found.");
                    }
                    break;

                case 7:
                    System.out.print("Enter key (username) to remove: ");
                    key = scanner.nextLine();
                    if (loginCredentials.containsKey(key)) {
                        loginCredentials.remove(key);
                        System.out.println("Key-value pair removed
successfully.");
                    } else {
                        System.out.println("Error: Key not found.");
                    }
                    break;

                case 8:
```

```
                    System.out.print("Enter key (username): ");
                    key = scanner.nextLine();
                    System.out.print("Enter value (password): ");
                    value = scanner.nextLine();
                    if (loginCredentials.remove(key, value)) {
                        System.out.println("Key-value pair removed
successfully.");
                    } else {
                        System.out.println("Error: Key-value pair not
found.");
                    }
                    break;

                case 9:
                    System.out.print("Enter value (password) to check: ");
                    value = scanner.nextLine();
                    if (loginCredentials.containsValue(value)) {
                        System.out.println("Value exists in the HashMap.");
                    } else {
                        System.out.println("Value does not exist in the
HashMap.");
                    }
                    break;

                case 10:
                    System.out.println("HashMap: " + loginCredentials);
                    break;

                case 11:
                    System.out.println("Exiting program...");
                    break;

                default:
                    System.out.println("Error: Invalid choice. Please try
again.");
                    break;
            }
        } while (choice != 11);

        scanner.close();
    }
}
```