1. WAP to take marks of 3 subjects from the User, compute total marks secured and percentage

App class

```
package com.main;
import java.util.Scanner;
import com.service.A;

public class App {
	public static void main(String[] args) {
		A a = new A();
		int n=3;
		Scanner sc = new Scanner(System.in);

		int[] arr = new int[10];
		for(int i=0;i<n;i++) {
			arr[i]=sc.nextInt();
		}
		double total = a.calculateTotal(arr);
		System.out.println("Total marks is " +total);

		double percent = a.computePercent(total,300.0);
		System.out.println("Percent is " +percent);
	}
}
```

A class

```
package com.main;

import java.util.Scanner;
import com.service.A;

public class App {
	public static void main(String[] args) {
		A a = new A();
		int n=3;
		Scanner sc = new Scanner(System.in);

		int[] arr = new int[10];
		for(int i=0;i<n;i++) {
			arr[i]=sc.nextInt();
		}

		double total = a.calculateTotal(arr);
		System.out.println("Total marks is " +total);

		double percent = a.computePercent(total,300.0);
		System.out.println("Percent is " +percent);
	}
}
```

---

2. In the above program, Compute Grade of the Student based on following criteria:
percent > 75: Grade A
percent > 60: Grade B
else : Grade C

App class

```
package com.main;

import java.util.Scanner;
import java.util.stream.IntStream;

import com.service.ComputeGrade;

public class App {
	public static void main(String[] args) {
		ComputeGrade a = new ComputeGrade();
		int n=3;
		Scanner sc = new Scanner(System.in);

		int[] arr = new int[10];
		for(int i=0;i<n;i++) {
			arr[i]=sc.nextInt();
		}

		double total = a.calculateTotal(arr);
		System.out.println("Total marks is " +total);

		double percent = a.computePercent(total,300.0);
		System.out.println("Percent is " +percent);

		String grade = a.computeGrade(percent);
		System.out.println("Grade is " +grade);
	}
}
```

ComputeGrade class

```
package com.service;

import java.util.stream.IntStream;

public class ComputeGrade {

	public int calculateTotal(int[] arr) {
		int sum = IntStream.of(arr).sum();
		return sum;
	}

	public double computePercent(double total, double d) {
		// TODO Auto-generated method stub
```

```
                double percent = (total*100) / d;
                return percent;

        }

        public String computeGrade(double percent) {
                // TODO Auto-generated method stub
//              if(percent >75)
//                      return "A";
//              if(percent >65)
//                      return "B";
//              else
//                      return "C";
                return percent > 75? "A" : percent>60? "B" : "C";
        }
}
```

Output:

```
67
89
98
Total marks is 254.0
Percent is 84.66666666666667
Grade is A
```

---

3. WAP to implement following Interface for implementing banking operations.

```
interface Deposit{
void deposit(Customer customer, double amount);
}
interface Withdrawal{
 double limit=50000;
 void withdraw(Customer customer, double amount)
}
```

Create Service classes(DepositService and WithdrawalService) to implement Deposit and Withdrawal interfaces respectively.
Call these service classes from App class and perform random deposits and withdrawals on at least 2 customers.

Customer class
id <int / Integer>
name <String>
balance <Double / double>
Note: Use Encapsulation to design Customer class.

Solution

App.java

```
package com.main;

import com.beans.Customer;
import com.exception.InsufficientFundsExp;
import com.exception.OverTheLimitExp;
import com.service.DepositService;
import com.service.WithdrawalService;

public class App {
	public static void main(String[] args) {
		Customer c1 = new Customer(80631,"Rakesh L",50000);
		Customer c2 = new Customer(80632,"Lokesh NS",65000);

		System.out.println("-----------Initial Amount-----------");
		System.out.println(c1);
		System.out.println(c2);

		DepositService ds = new DepositService();
		ds.deposit(c1, 28000);
		ds.deposit(c2, 43678);

		System.out.println("-----------Deposited Amount-----------");
		System.out.println(c1);
		System.out.println(c2);


		System.out.println("-----------Withdraw Amount-----------");
		WithdrawalService ws = new WithdrawalService();
		ws.withdraw(c1, 45000);
		ws.withdraw(c2, 47000);
		System.out.println(c1);
		System.out.println(c2);
	}
}
```

Customer.java

```
package com.beans;

public class Customer {
	private int id;
	private String name;
	private double balance;

	public Customer(int id, String name, double balance) {
		super();
		this.id = id;
		this.name = name;
		this.balance = balance;
```

```
	}

	public Customer() {
		super();
		// TODO Auto-generated constructor stub
	}

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getName() {
		return name;
	}

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

	public double getBalance() {
		return balance;
	}

	public void setBalance(double balance) {
		this.balance = balance;
	}

	@Override
	public String toString() {
		return "Customer [id=" + id + ", name=" + name + ", balance=" + balance
+ "]";
	}

}
```

Deposit.java (Interface)

```
package com.main;

import com.beans.Customer;

public interface Deposit {

	void deposit(Customer c, double amount);

}
```

Withdrawal.java (Interface)

```
package com.main;

import com.beans.Customer;
import com.exception.InsufficientFundsExp;
import com.exception.OverTheLimitExp;

public interface Withdrawal {
	double limit=50000;
	void withdraw(Customer c, double amount)
}
```

DepositService (Service)

```
package com.service;

import com.beans.Customer;
import com.main.Deposit;

public class DepositService implements Deposit{

	@Override
	public void deposit(Customer c, double amount) {
		c.setBalance(c.getBalance()+amount);
	}
}
```

WithdrawalService (Service)

```
package com.service;

import com.beans.Customer;
import com.exception.InsufficientFundsExp;
import com.exception.OverTheLimitExp;
import com.main.Withdrawal;

public class WithdrawalService implements Withdrawal{

	Customer c= new Customer();

	@Override
	public void withdraw(Customer c, double amount) throws OverTheLimitExp,
InsufficientFundsExp{
		c.setBalance(c.getBalance()-amount);

	}
}
```

Output:

```
-----------Initial Amount-----------
Customer [id=80631, name=Rakesh L, balance=30000.0]
Customer [id=80632, name=Lokesh NS, balance=65000.0]
-----------Deposited Amount-----------
Customer [id=80631, name=Rakesh L, balance=32000.0]
Customer [id=80632, name=Lokesh NS, balance=108678.0]
-----------Withdraw Amount-----------
Customer [id=80631, name=Rakesh L, balance=-13000.0]
Customer [id=80632, name=Lokesh NS, balance=61678.0]
```

---

4. Create 3 Employee Objects having following details
id:1
name: harry potter
city: London
salary: 85000

id:2
name: ronald weasley
city: surrey
salary: 75000

id:3
name: hermione granger
city: london
salary: 95000

Save these object in List and perform following operations: [Use either Comparable or Comparator Interface]
a. Sort as per salary ASC order
b. Sort as per salary DESC order

Solution

com.main

App.java

```
package com.main;

import java.util.ArrayList;

import com.beans.Employee;
import com.service.EmployeeService;
```

```
public class App {
	public static void main(String[] args) {
		Employee e1 = new Employee(1, "Harry Potter", "London", 85000);
		Employee e2 = new Employee(2, "Ronald weasley", "Surray", 75000);
		Employee e3 = new Employee(3, "Hermione Granger", "London",
95000);

		ArrayList<Employee> list = new ArrayList<>();

		list.add(e1);
		list.add(e2);
		list.add(e3);

		for(Employee e : list) {
			System.out.println(e);
		}

		EmployeeService employeeService = new EmployeeService();
		employeeService.sortASC(list);

		employeeService.sortDESC(list);
	}
}
```

com.beans

Employee.java

```
package com.beans;

public class Employee {
	private int id;
	private String name;
	private String city;
	private double salary;

	public Employee(int id, String name, String city, double salary) {
		super();
		this.id = id;
		this.name = name;
		this.city = city;
		this.salary = salary;
	}

	public Employee() {
		super();
		// TODO Auto-generated constructor stub
	}

	public int getId() {
		return id;
	}

	public void setId(int id) {
```

```
		this.id = id;
	}

	public String getName() {
		return name;
	}

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

	public String getCity() {
		return city;
	}

	public void setCity(String city) {
		this.city = city;
	}

	public double getSalary() {
		return salary;
	}

	public void setSalary(double salary) {
		this.salary = salary;
	}

	@Override
	public String toString() {
		return "Employee [id=" + id + ", name=" + name + ", city=" + city + ",
salary=" + salary + "]";
	}

}
```

com.service

EmployeeService.java

```
package com.service;

import java.util.ArrayList;
import java.util.Comparator;

import com.beans.Employee;

public class EmployeeService {

	public void sortASC(ArrayList<Employee> list) {
		// TODO Auto-generated method stub
		System.out.println("------------Employees Sorted By Ascending order By
Salary-----------");
		list.sort(Comparator.comparingDouble(Employee::getSalary));
		for(Employee e: list) {
```

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

        public void sortDESC(ArrayList<Employee> list) {
                // TODO Auto-generated method stub
                System.out.println("------------Employees Sorted By Desending order By
Salary-----------");
                list.sort(Comparator.comparingDouble(Employee::getSalary).reversed());
                for(Employee e: list) {
                        System.out.println(e);
                }
        }
}
```

Output:

```
Employee [id=1, name=Harry Potter, city=London, salary=85000.0]
Employee [id=2, name=Ronald weasley, city=Surray, salary=75000.0]
Employee [id=3, name=Hermione Granger, city=London, salary=95000.0]
------------Employees Sorted By Ascending order By Salary-----------
Employee [id=2, name=Ronald weasley, city=Surray, salary=75000.0]
Employee [id=1, name=Harry Potter, city=London, salary=85000.0]
Employee [id=3, name=Hermione Granger, city=London, salary=95000.0]
------------Employees Sorted By Desending order By Salary-----------
Employee [id=3, name=Hermione Granger, city=London, salary=95000.0]
Employee [id=1, name=Harry Potter, city=London, salary=85000.0]
Employee [id=2, name=Ronald weasley, city=Surray, salary=75000.0]
```

---

5. In the above case study, filter employees based on following criteria:

a. Display employees having salary<80000
b. Display employees living in city 'london'

Solution:

```
com.main
--------
App.java
--------
package com.main;

import java.util.ArrayList;

import com.beans.Employee;
import com.service.EmployeeService;

public class App {
        public static void main(String[] args) {
                Employee e1 = new Employee(1, "Harry Potter", "London", 85000);
                Employee e2 = new Employee(2, "Ronald weasley", "Surray", 75000);
```

```
		Employee e3 = new Employee(3, "Hermione Granger", "London",
95000);

		ArrayList<Employee> list = new ArrayList<>();

		list.add(e1);
		list.add(e2);
		list.add(e3);

		for(Employee e : list) {
			System.out.println(e);
		}

		EmployeeService employeeService = new EmployeeService();
		employeeService.filteringSalary(list);

		employeeService.filteringCity(list);
	}
}
```

com.beans

Employee.java

```
package com.beans;

public class Employee {
	private int id;
	private String name;
	private String city;
	private double salary;

	public Employee(int id, String name, String city, double salary) {
		super();
		this.id = id;
		this.name = name;
		this.city = city;
		this.salary = salary;
	}

	public Employee() {
		super();
		// TODO Auto-generated constructor stub
	}

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getName() {
```

```
		return name;
	}

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

	public String getCity() {
		return city;
	}

	public void setCity(String city) {
		this.city = city;
	}

	public double getSalary() {
		return salary;
	}

	public void setSalary(double salary) {
		this.salary = salary;
	}

	@Override
	public String toString() {
		return "Employee [id=" + id + ", name=" + name + ", city=" + city + ",
salary=" + salary + "]";
	}

}
```

com.service

EmployeeService.java

```
package com.service;

import java.util.ArrayList;

import com.beans.Employee;

public class EmployeeService {

	public void filteringSalary(ArrayList<Employee> list) {
		// TODO Auto-generated method stub
		System.out.println("------------Employees Salary Greater Than 80000------
-----");
		for(Employee e: list) {
			if(e.getSalary()>80000) {
				System.out.println(e);
			}
		}
	}
```

```
	public void filteringCity(ArrayList<Employee> list) {
		// TODO Auto-generated method stub
		System.out.println("------------Employees who belongs to City London-----
------");
		for(Employee e: list) {
			if(e.getCity()=="London") {
				System.out.println(e);
			}
		}


	}

}
```

Output:

```
Employee [id=1, name=Harry Potter, city=London, salary=85000.0]
Employee [id=2, name=Ronald weasley, city=Surray, salary=75000.0]
Employee [id=3, name=Hermione Granger, city=London, salary=95000.0]
------------Employees Salary Greater than 80000-----------
Employee [id=1, name=Harry Potter, city=London, salary=85000.0]
Employee [id=3, name=Hermione Granger, city=London, salary=95000.0]
------------Employees who belongs to City London-----------
Employee [id=1, name=Harry Potter, city=London, salary=85000.0]
Employee [id=3, name=Hermione Granger, city=London, salary=95000.0]
```

---

6. In case study 3 above, perform following validations using self defined exceptions.
InsufficientFundsException:

if amount > balance of the customer, throw this exception with the message "Insufficient Funds"
OverTheLimitException:

if amount > 50000 during withdrawal, throw this exception with the message "Limit 50000
Exceeded"

Note: Both the exceptions should be checked exceptions.

Solution:

App.java

```
package com.main;
import com.beans.Customer;
import com.exception.InsufficientFundsExp;
import com.exception.OverTheLimitExp;
import com.service.DepositService;
import com.service.WithdrawalService;

public class App {
	public static void main(String[] args) {
		Customer c1 = new Customer(80631,"Rakesh L",50000);
		Customer c2 = new Customer(80632,"Lokesh NS",65000);

		System.out.println("-----------Initial Amount-----------");
		System.out.println(c1);
		System.out.println(c2);

		DepositService ds = new DepositService();
		ds.deposit(c1, 28000);
		ds.deposit(c2, 43678);

		System.out.println("-----------Deposited Amount-----------");
		System.out.println(c1);
		System.out.println(c2);

		System.out.println("-----------Withdraw Amount-----------");
		WithdrawalService ws = new WithdrawalService();
		try {
			ws.withdraw(c1, 45000);
			ws.withdraw(c2, 67657);
		}
		catch (InsufficientFundsExp e) {
			System.out.println(e.getMessage());
		}
		catch (OverTheLimitExp e) {
			System.out.println(e.getMessage());
		}
		System.out.println(c1);
		System.out.println(c2);
		System.out.println("thank you");
	}
}
```

Customer.java

```
package com.beans;

public class Customer {
	private int id;
	private String name;
	private double balance;
```

```
	public Customer(int id, String name, double balance) {
		super();
		this.id = id;
		this.name = name;
		this.balance = balance;
	}

	public Customer() {
		super();
		// TODO Auto-generated constructor stub
	}

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getName() {
		return name;
	}

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

	public double getBalance() {
		return balance;
	}

	public void setBalance(double balance) {
		this.balance = balance;
	}

	@Override
	public String toString() {
		return "Customer [id=" + id + ", name=" + name + ", balance=" + balance
+ "]";
	}

}
```

<u>Deposit.java (Interface)</u>

```
package com.main;

import com.beans.Customer;

public interface Deposit {

	void deposit(Customer c, double amount);
```

```
}
```

Withdrawal.java (Interface)

```
package com.main;

import com.beans.Customer;
import com.exception.InsufficientFundsExp;
import com.exception.OverTheLimitExp;

public interface Withdrawal {
	double limit=50000;
	void withdraw(Customer c, double amount)
		throws InsufficientFundsExp, OverTheLimitExp;
}
```

DepositService (Service)

```
package com.service;

import com.beans.Customer;
import com.main.Deposit;

public class DepositService implements Deposit{

	@Override
	public void deposit(Customer c, double amount) {
		c.setBalance(c.getBalance()+amount);
	}
}
```

WithdrawalService (Service)

```
package com.service;

import com.beans.Customer;
import com.exception.InsufficientFundsExp;
import com.exception.OverTheLimitExp;
import com.main.Withdrawal;

public class WithdrawalService implements Withdrawal{

	Customer c= new Customer();

	@Override
	public void withdraw(Customer c, double amount) throws OverTheLimitExp,
InsufficientFundsExp{

		if(amount > Withdrawal.limit)
			throw new OverTheLimitExp("Withdrawal amount cannot be
more than " + Withdrawal.limit + " ID = " +c.getId());
```

```
                if(amount > c.getBalance())
                        throw new InsufficientFundsExp("Please Enter the amount that
is less than or equal to your balance");
                c.setBalance(c.getBalance()-amount);

        }
}
```

InsufficientFundsExp.java (Exception)

```
package com.exception;

public class InsufficientFundsExp extends Exception{

        private static final long serialVersionUID = 1L;
        private String message;

        public InsufficientFundsExp(String message) {
                this.message = message;
        }

        public String getMessage(){
                return message;
        }

}
```

OverTheLimitExp.java (Exception)

```
package com.exception;

public class OverTheLimitExp extends Exception{

        private static final long serialVersionUID = 1L;
        private String message;

        public OverTheLimitExp(String message) {
                this.message = message;
        }

        public String getMessage(){
                return message;
        }

}
```

Output:

```
-----------Initial Amount-----------
Customer [id=80631, name=Rakesh L, balance=30000.0]
Customer [id=80632, name=Lokesh NS, balance=65000.0]
-----------Deposited Amount-----------
Customer [id=80631, name=Rakesh L, balance=32000.0]
Customer [id=80632, name=Lokesh NS, balance=108678.0]
```

```
-----------Withdraw Amount for InsufficientFundsExp-----------
Insufficient Funds
Customer [id=80631, name=Rakesh L, balance=32000.0]
Customer [id=80632, name=Lokesh NS, balance=108678.0]
thank you
-----------Withdraw Amount for OverTheLimitExp-----------
Limit 50000 Exceeded
Customer [id=80631, name=Rakesh L, balance=32000.0]
Customer [id=80632, name=Lokesh NS, balance=108678.0]
thank you
```

---

7. Login System using Map.

Save 5 username/passwords in HashMap with username as key and password as value.

Take username/password as Input from the User and check if they are valid against the entries of HashMap.

Solution:

<u>App.java</u>

```
package com.main;

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

public class App {
	public static void main(String[] args) {
		Map<String,String> map = new HashMap<>();

		map.put("rakesh", "lokesh");
		map.put("shiva", "09051996");
		map.put("bharathi", "basavaraj");
		map.put("lokesh", "1999");
		map.put("yashas", "braj");

		System.out.println("-----------Login-----------");
		/* take username from the user and verify if its present in the map.  */
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter the Username: ");
		String username = sc.next();

		if(!map.containsKey(username)) {
			System.out.println("Invalid Username");
			sc.close();
			return;
```

```
                }

                System.out.println("Enter the password: ");
                String password=sc.next();

                String passMap = map.get(username);

                if(!password.equals(passMap)) {
                        System.out.println("Invalid password");
                        sc.close();
                        return;
                }

                System.out.println("Login Success, Welcome " + username);
                sc.close();
        }
}
```

Output:

op-1

```
-----------Login-----------
Enter the Username:
rakesh
Enter the password:
lokesh
Login Success, Welcome rakesh
```

op-2

```
-----------Login-----------
Enter the Username:
raju
Invalid Username
```

op-3

```
-----------Login-----------
Enter the Username:
lokesh
Enter the password:
rakesh
Invalid password
```

---

8. Case Study: Menu Driven Program using JDBC API

Create a table product(id,name,price,description) in MySql DB.

Case 1:
Write a program to Insert the records in the table using JDBC API. Note: Taking input from User.
Case 2: Display all products from the DB

Case 3: Delete product based on id
Case 0: exit

Solution:

App.java

```
package com.main;

import java.util.List;
import java.util.Scanner;

import com.beans.Product;
import com.service.ProductService;

public class App {
	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		ProductService productService = new ProductService();

		System.out.println("1. Insert product");
	System.out.println("2. Delete product by ID");
	System.out.println("3. Display all product");
		System.out.println("0. Exit");
	int input = sc.nextInt();

	switch(input) {
	case 1:
		System.out.println("Enter product Details");
		System.out.println("Product Name: ");
		String name = sc.next();
		System.out.println("Prouct Price: ");
	double price = sc.nextDouble();
	System.out.println("Product Description: ");
	String description = sc.next();

	Product product = new Product();
	product.setName(name);
	product.setPrice(price);
	product.setDescription(description);
	productService.insert(product);

	productService.insert(product);
	System.out.println("Employee Record Inserted.. ");

	case 2:
		System.out.println("######## Delete Product #########");
			System.out.println("Enter the ID of Product to delete");
			int id = sc.nextInt();
			productService.deleteProduct(id);
			System.out.println("Product record deleted...");
			break;
```

```
            case 3:
                System.out.println("######### Display all employees ########");
                        List<Product> list = productService.fetchEmployees();
                        for(Product p : list) {
                                System.out.println(p);
                        }
                        break;
            case 0:
                System.out.println("Invalid Entery");
                break;
            }
        }
}
```

Product.java

```
package com.beans;

public class Product {

        private int id;
        private String name;
        private double price;
        private String description;

        public Product(int id, String name, double price, String description) {
                super();
                this.id = id;
                this.name = name;
                this.price = price;
                this.description = description;
        }

        public Product() {
                super();
                // TODO Auto-generated constructor stub
        }

        public int getId() {
                return id;
        }

        public void setId(int id) {
                this.id = id;
        }

        public String getName() {
                return name;
        }

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

```
	public double getPrice() {
		return price;
	}

	public void setPrice(double price) {
		this.price = price;
	}

	public String getDescription() {
		return description;
	}

	public void setDescription(String description) {
		this.description = description;
	}

	@Override
	public String toString() {
		return "Product [id=" + id + ", name=" + name + ", price=" + price + ",
description=" + description + "]";
	}

}
```

ProductService.java

```
package com.service;

import java.util.List;

import com.beans.Product;

public class ProductService {
	ProductDB productDB = new ProductDB();


   public void insert(final Product product) {
      productDB.insert(product);
   }


	public void deleteProduct(int id) {
		// TODO Auto-generated method stub
		productDB.deleteProduct(id);
	}


	public List<Product> fetchEmployees() {

		return productDB.fetchEmployees();
	}
}
```

ProductDB.java

```
package com.service;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import com.beans.Product;

public class ProductDB {
	Connection con;
	public void dbConnect() {

		try {
			Class.forName("com.mysql.cj.jdbc.Driver");

		} catch (ClassNotFoundException e) {
			e.printStackTrace();
		}

		try {
			con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/myproductassigment_80631"
								, "root", "Password123");

		} catch (SQLException e) {
			e.printStackTrace();
		}
	}

	public void insert(Product product) {
		// TODO Auto-generated method stub
		dbConnect();
		String sql="insert into product(name,price,description) values (?,?,?)";
		try {
			PreparedStatement pstmt = con.prepareStatement(sql);

			pstmt.setString(1, product.getName());
			pstmt.setDouble(2, product.getPrice());
			pstmt.setString(3, product.getDescription());


			pstmt.executeUpdate();
		} catch (SQLException e) {
			e.printStackTrace();
		}
		dbClose();
	}
```

```
	public void deleteProduct(int id) {
		// TODO Auto-generated method stub
		dbConnect();
		String sql="delete from product where id=?";
		try {
			PreparedStatement pstmt = con.prepareStatement(sql);
			pstmt.setInt(1, id);
			pstmt.executeUpdate();
		} catch (SQLException e) {
			e.printStackTrace();
		}
		dbClose();
	}
	public List<Product> fetchEmployees() {
		// TODO Auto-generated method stub
		dbConnect();
		String sql="select * from product";

		List<Product> list = new ArrayList<>();
		try {
			PreparedStatement pstmt = con.prepareStatement(sql);
			ResultSet rst = pstmt.executeQuery();
			while(rst.next()) {
				Product p = new Product();
				int id = rst.getInt("id");
				String name = rst.getString("name");
				double price = rst.getDouble("price");
				String description = rst.getString("description");
				p.setId(id);
				p.setName(name);
				p.setPrice(price);
				p.setDescription(description);
				list.add(p);
			}
		} catch (SQLException p) {

			p.printStackTrace();
		}
		dbClose();
		return list;
	}
	public void dbClose() {
		try {
			con.close();

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

Database Query

create database myproductassigment_80631;

use myproductassigment_80631;

create table product(id int primary key auto_increment, name varchar(255) NOT NULL, price double default 0, description varchar(255));

describe product;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
name	varchar(255)	NO		NULL	
price	double	YES		0	
description	varchar(255)	YES		NULL	

Result 10

select * from product;

drop table product;

Output

case 1: Insert the records in the table

1. Insert product
2. Delete product by ID
3. Display all product
0. Exit

1

Enter product Details
Product Name:
Apple_Iphone_SE

Prouct Price:
18000

Product Description:
8GB_RAM,16GB_Memory,12'TFT

Product Record Inserted..

```
Markers  Properties  Servers  Console  Data Sour
App (7) [Java Application] C:\Program Files\Java\jre1.8.0_281\bin\javaw.e
1. Insert product
2. Delete product by ID
3. Display all product
0. Exit
1
Enter product Details
Product Name:
Apple_Iphone_SE
Prouct Price:
18000
Product Description:
8GB_RAM,16GB_Memory,12'TFT
Product Record Inserted..
```

case 2: Display all products from the DB

1. Insert product
2. Delete product by ID
3. Display all product
0. Exit

3

######### Display all employees ########
Product [id=1, name=Oppo_A12_Mobile_Phone, price=14000.0, description=8GB_RAM,64GB_Memory,15'TFT]
Product [id=2, name=HP_Laser_L18_Laptop, price=54000.0, description=8GB_RAM,1TB_HDD,18'LED]
Product [id=3, name=Lenovo_Think_Pad, price=72000.0, description=16GB_RAM,256GB_SSD,17'LED]
Product [id=4, name=Samsung_S16_Mobile_Phone, price=13000.0, description=8GB_RAM,32GB_Memory,17'TFT]
Product [id=5, name=Apple_Iphone_SE, price=18000.0, description=8GB_RAM,16GB_Memory,12'TFT]
Product [id=6, name=Dell_Latitute_Laptop, price=35000.0, description=8GB_RAM,1TB_HDD,15'LED]
Product [id=7, name=Poco_M2_Pro, price=22000.0, description=6GB_RAM,64GB_Memory_16'TFT]

```
Markers  Properties  Servers  Console  Data Source Explorer  Snippets
<terminated> App (7) [Java Application] C:\Program Files\Java\jre1.8.0_281\bin\javaw.exe (Sep 20, 2022, 11:20:17 PM)
1. Insert product
2. Delete product by ID
3. Display all product
0. Exit
3
######## Display all employees ########
Product [id=1, name=Oppo_A12_Mobile_Phone, price=14000.0, description=8GB_RAM,64GB_Memory,15'TFT]
Product [id=2, name=HP_Laser_L18_Laptop, price=54000.0, description=8GB_RAM,1TB_HDD,18'LED]
Product [id=3, name=Lenovo_Think_Pad, price=72000.0, description=16GB_RAM,256GB_SSD,17'LED]
Product [id=4, name=Samsung_S16_Mobile_Phone, price=13000.0, description=8GB_RAM,32GB_Memory,17'TFT]
Product [id=5, name=Apple_Iphone_SE, price=18000.0, description=8GB_RAM,16GB_Memory,12'TFT]
Product [id=6, name=Dell_Latitute_Laptop, price=35000.0, description=8GB_RAM,1TB_HDD,15'LED]
Product [id=7, name=Poco_M2_Pro, price=22000.0, description=6GB_RAM,64GB_Memory_16'TFT]
```

Result Grid | Filter Rows: | Edit: | Export/Import:

id	name	price	description
1	Oppo_A12_Mobile_Phone	14000	8GB_RAM,64GB_Memory,15'TFT
2	HP_Laser_L18_Laptop	54000	8GB_RAM,1TB_HDD,18'LED
3	Lenovo_Think_Pad	72000	16GB_RAM,256GB_SSD,17'LED
4	Samsung_S16_Mobile_Phone	13000	8GB_RAM,32GB_Memory,17'TFT
5	Apple_Iphone_SE	18000	8GB_RAM,16GB_Memory,12'TFT
6	Dell_Latitute_Laptop	35000	8GB_RAM,1TB_HDD,15'LED
7	Poco_M2_Pro	22000	6GB_RAM,64GB_Memory_16'TFT
NULL	NULL	NULL	NULL

product 14

case 3: Delete product based on id

1. Insert product
2. Delete product by ID
3. Display all product
0. Exit

2
######## Delete Product #########
Enter the ID of Product to delete

7
Product record deleted...

Markers | Properties | Servers | Console

```
<terminated> App (7) [Java Application] C:\Program Files\Ja
1. Insert product
2. Delete product by ID
3. Display all product
0. Exit
2
######## Delete Product #########
Enter the ID of Product to delete
7
Product record deleted...
```

Database Table after Deletion (Record ID =7 got deleted)

Result Grid | Filter Rows: | Edit: | Export/Import:

id	name	price	description
1	Oppo_A12_Mobile_Phone	14000	8GB_RAM,64GB_Memory,15'TFT
2	HP_Laser_L18_Laptop	54000	8GB_RAM,1TB_HDD,18'LED
3	Lenovo_Think_Pad	72000	16GB_RAM,256GB_SSD,17'LED
4	Samsung_S16_Mobile_Phone	13000	8GB_RAM,32GB_Memory,17'TFT
5	Apple_Iphone_SE	18000	8GB_RAM,16GB_Memory,12'TFT
6	Dell_Latitute_Laptop	35000	8GB_RAM,1TB_HDD,15'LED
NULL	NULL	NULL	NULL

product 16

Case 0: exit

1. Insert product
2. Delete product by ID
3. Display all product
0. Exit

0
Invalid Entery

```
Markers   Properties   Servers
<terminated> App (7) [Java Application] C:\Pr
1. Insert product
2. Delete product by ID
3. Display all product
0. Exit
0
Invalid Entery
```

Project Explore