Name: Chovatiya Mansvi Vijaybhai
Roll no:CE076
A4

Lab 8

Programming Exercise:

1. Write a Java program to create an Employee class having attributes empId and salary. Sort the employee objects using following two different Comparator classes and display the sorted list:
    a. by employee id
    b. by salary

```
import java.util.*;
class employee
{
   int empId;
   int salary;
   employee(int empId,int salary)
   {
       this.empId=empId;
       this.salary=salary;
   }
   public String toString()
   {
       return "Id:"+empId+" salary:"+salary;
   }
}
class sortById implements Comparator<employee>
{
   public int compare(employee e1,employee e2)
   {
       return e1.empId-e2.empId;
   }
}
class sortBySalary implements Comparator<employee>
{
   public int compare(employee e1,employee e2)
   {
       return e1.salary-e2.salary;
   }
}
public class program
{
   public static void main(String []args)
   {
```

```
        List<employee> list=new ArrayList<>();
        list.add(new employee(101,68000));
        list.add(new employee(407,50000));
        list.add(new employee(268,89000));

        Collections.sort(list,new sortById());
        System.out.println("sort by id:"+list);

        Collections.sort(list,new sortBySalary());
        System.out.println("sort by salary"+list);
    }
}
```

Output:

```
user1@celab4-HP-ProDesk-400-G7-Microtower-PC:~/IdeaProjects/javaprogram/src$ javac program.java
user1@celab4-HP-ProDesk-400-G7-Microtower-PC:~/IdeaProjects/javaprogram/src$ java program
sort by id:[Id:101 salary:68000, Id:268 salary:89000, Id:407 salary:50000]
sort by salary[Id:407 salary:50000, Id:101 salary:68000, Id:268 salary:89000]
```

2. Write a Java program to create a class City having attributes cityName and population. Implement the Comparable interface to sort city objects in descending order of population and display the sorted list.

```
import java.util.*;
import java.lang.*;
class City implements Comparable<City>
{
    String cityname;
    int population;
    City(String cityname,int population)
    {
        this.cityname=cityname;
        this.population=population;
    }
    @Override
    public int compareTo(City c)
    {
        return this.population-c.population;
    }
    @Override
    public String toString()
    {
        return "name:"+cityname+" population:"+population;
    }
}
public class program
```

```
{
   public static void main(String []args)
   {
        List<City> list=new ArrayList<>();
        list.add(new City("Delhi",17000868));
        list.add(new City("Mumbai",20487864));
        list.add(new City("surat",9957842));
        list.add(new City("nadiad",265134));

        Collections.sort(list);
        System.out.println("sort by population:"+list);
   }
}
```

Output:

```
user1@celab4-HP-ProDesk-400-G7-Microtower-PC:~/IdeaProjects/javaprogram/src$ javac program.java

user1@celab4-HP-ProDesk-400-G7-Microtower-PC:~/IdeaProjects/javaprogram/src$ java program
sort by population:[name:nadiad population:265134, name:surat population:9957842, name:Delhi population:17000868, name:Mumbai population:20487864]
```

3. Consider an Online Order Processing System developed using the Spring Framework. The application contains the following components:
   - **PaymentService**: An interface that contains a method to make a payment. Two classes implement the PaymentService interface:
     - **CreditCardPaymentService**
     - **PaypalPaymentService**
   - **InventoryService**: Used to check whether a product is available in stock.
   - **OrderService**: Uses PaymentService and InventoryService to place an order.

   Write a Spring-based Java application that meets the requirements below:
   1. Use annotation-based configuration.

```
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;

@Configuration
@ComponentScan(basePackages = "com.example")
public class AppConfig {
}
```

   2. Use constructor-based dependency injection.

   3. Mark CreditCardPaymentService as the default payment service using the @Primary annotation.

4. Use the @Qualifier annotation to explicitly inject PaypalPaymentService in another service.

5. Use the following main class to test the application.

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class MainApp {
   public static void main(String[] args) {
      ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);

      OrderService orderService = context.getBean(OrderService.class);
      orderService.placeOrder("P101", 1000.0);
      PaypalOrderService paypalOrderService = context.getBean(PaypalOrderService.class);
      paypalOrderService.placeOrder(500.0);
   }
}
```

Expected Output:

```
Checking availability for product: P101
Credit Card payment processed: 1000.0
Order placed successfully.

PayPal payment processed: 500.0
PayPal order placed successfully.
```

InventoryService.java

```java
package com.order.online.orderonline;

import org.springframework.stereotype.Component;

@Component
public class InventoryService {

   public boolean checkAvailability(String productId) {
       System.out.println("Checking availability for product: " + productId);
       return true;
   }
}
```

MainApp.java

```
package com.order.online.orderonline;

import org.springframework.context.ApplicationContext;
import

org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class MainApp {

   public static void main(String[] args) {

       ApplicationContext context =
               new AnnotationConfigApplicationContext(AppConfig.class);

       OrderService orderService = context.getBean(OrderService.class);
       orderService.placeOrder("P101", 1000.0);

       PaypalOrderService paypalOrderService =
               context.getBean(PaypalOrderService.class);
       paypalOrderService.placeOrder(500.0);
   }
}
```

OrderService.java

```
package com.order.online.orderonline;

import org.springframework.stereotype.Component;
import com.order.online.payment.PaymentService;

@Component
public class OrderService {

   private final PaymentService paymentService;
   private final InventoryService inventoryService;

   public OrderService(PaymentService paymentService,
                       InventoryService inventoryService) {
       this.paymentService = paymentService;
       this.inventoryService = inventoryService;
   }

   public void placeOrder(String productId, double amount) {
       if (inventoryService.checkAvailability(productId)) {
           paymentService.makePayment(amount);
           System.out.println("Order placed successfully.");
       }
   }
}
```

PaypalOrderService.java

```
package com.order.online.orderonline;

import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Component;
import com.order.online.payment.PaymentService;

@Component
public class PaypalOrderService {

   private final PaymentService paymentService;

   public PaypalOrderService(
           @Qualifier("paypalPaymentService") PaymentService paymentService)
   {
       this.paymentService = paymentService;
   }

   public void placeOrder(double amount) {
       paymentService.makePayment(amount);
       System.out.println("PayPal order placed successfully.");
   }
}
```

PaymentService.java

```
package com.order.online.payment;

public interface PaymentService {
   void makePayment(double amount);
}

CreditCardPaymentService.java
package com.order.online.payment;

import org.springframework.context.annotation.Primary;
import org.springframework.stereotype.Component;

@Component
@Primary
public class CreditCardPaymentService implements PaymentService{
   @Override
   public void makePayment(double amount) {
       System.out.println("Credit Card payment processed: " + amount);
   }
}
```

PaypalPaymentService.java

```java
package com.order.online.payment;
import org.springframework.context.annotation.Primary;
import org.springframework.stereotype.Component;
@Component
public class PaypalPaymentService implements PaymentService{
   @Override
   public void makePayment(double amount) {
       System.out.println("Paypal payment processed: " + amount);
   }
}
```

Output:-

```
Checking availability for product: P101
Credit Card payment processed: 1000.0
Order placed successfully.
Paypal payment processed: 500.0
PayPal order placed successfully.
```