# Semester Test

2nd Semester

Student Name : Fei Gu

CPR-No : 1206874011

Email: feix0033@easv365.dk

Class : DMU 20

GitHub repository: https://github.com/evensnachi/InternTest.git

# JAVA

## Collections, Searching, Sorting

### Q1 : In which of the labelled positions (A-H) is the number 17 to be placed, if inserted correctly?

Answer: The label **C** is the place for number 17.

### Q2 : In the worst case, how many elements do we need to check (compare with) if searching for a specific value?

Answer: **3** numbers should be check.

( If we didn't placed the number 17 from last question , otherwise that should be **4** numbers in the worst case. )

---

## Terminology

### Q3 : Using a linear search algorithm in an array of N elements (starting in one end, and finishing in the other). With what Big-Oh expression would you describe the efficiency of this search algorithm?

Answer : O(n)

## Q4 : What is an anonymous inner-class and does it have any advantages compared to using a normal Class.

Answer :

The anonymous inner-class is the temporary class without initialised  class name. Therefore the class can only be used one times in the class.

Advantage: no mistake instance. The class is be protected.  Only one instance so the member variably will not be changed.

## Q5 : Suppose C is a class that implements the interfaces I and J.The following is given:

```java
I i = new C();
```

Which of the following statements will throw an exception,

A: `C c = (C) i`;

B: `J j = (J) i`;

C: `I = (I) null`;

Answer : **B**

## Q6 : What is an instance method and how does it differ from a static method? Describe in your own words.Give a code example.

Answer :

The instance method is the method be created in the class. When we need use the method, we should instance the class first and then invoke the method by formate:

`Class instaceName = new Class;`

`instaceName.methodName();`

The static method is a static method be created in the  class and it don’t need to be instace to invoke the method. It just need use by this `ClassName.methodName();`

## JavaFX

### Q7 : Create a calculator, which can add two whole numbers. Use databinding for adding the two input values.

```java
    public static void main(String[] args) {
        launch(args);
    }


    @Override
    public void start(Stage primaryStage) {
        Label label = new Label("Calculator");

        TextField textField1 = new TextField();
        textField1.setLayoutX(100);
        textField1.setLayoutX(2);

        Label label1 = new Label("+");
        TextField textField2 = new TextField();

        String result = textField1.getText() + textField2.getText();
        Label label2 = new Label("=");



        Label label3 = new Label();

        HBox hBox = new HBox(textField1, label1, textField2, label2,label3);
        hBox.setLayoutX(2);
        hBox.setLayoutY(30);

        Pane pane = new Pane(label,hBox);
        Scene scene = new Scene(pane,600,400);
        primaryStage.setScene(scene);
        primaryStage.setTitle("Calculator");
        primaryStage.show();

    }
}
```

## Collections, Recursion

### Q8 : Create a class named “Customer”, which contains the information of the test customers. (See table) Add the four customers to a List.

Answer :

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

        /*
          * Q8
          */
        System.out.println("************* Q8 ***************");

        Customer customer1 = new Customer("Hugo", 95, 55);
        Customer customer2 = new Customer("Svenja", 35, 150);
        Customer customer3 = new Customer("Antonio", 45, 250);
        Customer customer4 = new Customer("Christina-Antoinette", 19, 400);

        List arr = new ArrayList();
        arr.add(customer1);
        arr.add(customer2);
        arr.add(customer3);
        arr.add(customer4);

        for (Object o : arr) {
            System.out.println(o);
        }
    }
}

public class Customer {
    String name;
    int age;
    int price;

    public Customer() {
    }

    public Customer(String name, int age, int price) {
        this.name = name;
        this.age = age;
        this.price = price;
    }

    public String getName() {
        return name;
    }

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

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public int getPrice() {
        return price;
    }

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

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Customer customer = (Customer) o;
        return age == customer.age && price == customer.price && Objects.equals(name, customer.name);
    }

    @Override
    public int hashCode() {
        return Objects.hash(name, age, price);
    }

    @Override
    public String toString() {
        return "Customer{" +
                "name='" + name + '\'' +
                ", age=" + age +
                ", price=" + price +
                '}';
    }
}

```

```
```

**Q9 : Implement the necessary change to enable the list of Customers in being sorted by the method “sort” of the class “java.util.Collections”.The list must be sorted ascending in regards to the property Age.**

Answer :

```java
public class CollectionsRecursion {
    public static void main(String[] args) {
        /*
         * Q9
         */

        Collections.sort(arr);
        for (Object o : arr) {
            System.out.println(o);
        }

    }
}
public class Customer implements Comparable{
    @Override
    public int compareTo(Object o) {
        if(o instanceof Customer){
            Customer customer = (Customer) o;
            if(this.age > customer.age){
                return 1;
            }
            else if(this.age < customer.age){
                return -1;
            }
            else {
                return 0;
            }
        }
        throw new RuntimeException("the date is incorrect");
    }
}
```

**Q10 : Create a recursive method which can print the content of the list in the following format : “Name, Age (Years), Price (Kr)” Like : Hugo, 95 (Years), 55 (Kr)**

## Unit Testing

**Q11 : Write the plain Java code (inside the curly braces) so the method “smallest” above fulfils the requirements stated.**

```java
public static int smallest(int a, int b, int c){
        if(a < b){
            if(a < c){
                return a;
            }
        }else {
            if(b < c){
                return b;
            }
        }
        return c;
    }

```

**Q12 : Write JUnit test code (in two methods using asserts) for the above method. You must test these two set of values:**

```java
 @Test
    public void test(){
        int actual= smallest(2, 3, 4);
        int expected = 2;
        assertEquals(expected,actual);
        System.out.println("actual:  " + actual + " expected:  " + expected
);
    }

 @Test
    public void test2(){
        int actual= smallest(1, -3,0);
        int expected = -3;
        assertEquals(expected,actual);
        System.out.println("actual:  " + actual + " expected:  " + expected
);

```

```
    }
```

# Database and applications

## Batch Queries

### Q13 : Write a batch query that creates the database structure given above. This batch must create the two tables and also insert at least three records in each table.

```java
public static void main(String[] args) {
        try {
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
            Connection con = DriverManager.getConnection("jdbc:sqlserver://localhost:1433;databaseName=DB_Bikes","sa","Sa@123456");
            System.out.println(con);

            PreparedStatement ps = con.prepareStatement("INSERT INTO DB_Bikes.tbl_bike VALUES(?,?,?,?)");
            PreparedStatement ps1 = con.prepareStatement("INSERT INTO DB_Bikes.tbl_category(category_name) VALUES(?)");

            ps.setInt(1,001);
            ps.setString(2,"R");
            SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy-MM-dd");
            Date date = simpleDateFormat.parse("2000-01-01");
            ps.setDate(3,new java.sql.Date(date.getTime()));
            ps.setString(4,"Danmark");

            ps1.setString(1,"Road bike");

            ps.addBatch();

            ps.setInt(1,002);
            ps.setString(2,"M");
            SimpleDateFormat simpleDateFormat1 = new SimpleDateFormat("yyyy-MM-dd");
            Date date1 = simpleDateFormat.parse("2010-01-01");
            ps.setDate(3,new java.sql.Date(date1.getTime()));
            ps.setString(4,"China");

            ps1.setString(1,"Mountain bike");

            ps.addBatch();


            ps.setInt(1,003);
            ps.setString(2,"F");
            SimpleDateFormat simpleDateFormat2 = new SimpleDateFormat("yyyy-MM-dd");
            Date date2 = simpleDateFormat.parse("2020-01-01");
            ps.setDate(3,new java.sql.Date(date2.getTime()));
            ps.setString(4,"German");

            ps1.setString(1,"Flodring bike");

            ps.addBatch();



            int[] a = ps.executeBatch();

            for (int i = 0; i < a.length; i++) {
                System.out.println("The number is : " + a[i]);
            }
        }catch (Exception e){
            e.printStackTrace();
        }
    }
```

## JDBC

**Q14 : Write this method in a Java class, using JDBC and the interface PreparedStatement: public static int insertCategoryRecord(String bikeType, String categoryName) {...} The method must insert a new record in the tbl_category and return the number of records changed in the database.**

```java
public static int insertCategoryRecord(String bikeType, String categoryName) throws SQLException {

        Connection con = null;
        PreparedStatement ps = null;

        try {
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
            con = DriverManager.getConnection("jdbc:sqlserver://localhost:1433;databaseName=DB_Bikes","sa","Sa@123456");
            System.out.println(con);

            ps = con.prepareStatement("INSERT INTO tbl_category VALUES(?,?)");
            ps.setString(1,'"' + bikeType + '"');
            ps.setString(2,'"' + categoryName + '"');

            ResultSet rs = ps.executeQuery();

            ResultSetMetaData rsmd = rs.getMetaData();
            int columns = rsmd.getColumnCount();
            return columns;

        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if(ps != null){
                ps.close();
            }
            if(con != null){
                con.close();
            }
        }
        return 0;

    }
}
```