

Core Java

Topic: String

Problems:

1) Given string "Algorithms", return "go" and "Algo" using substring

```
J Main.java x
J Main.java > Main > main(String[])
1 public class Main {
    Run | Debug
2     public static void main(String[] args) {
3         String str = "Algorithms";
4
5         String substring1 = str.substring(2, 4);
6         System.out.println(substring1);
7
8         String substring2 = str.substring(0, 4);
9         System.out.println(substring2);
10    }
11 }
12
13
```

```
go
Algo
```

2) Given two strings, compare if these two strings are equal by comparing each character

```
public class stringEqual {
    Run | Debug
    public static void main(String[] args) {
        String str1 = "Algorithms";
        String str2 = "Algorithxs";

        boolean isEqual = compareStrings(str1, str2);
        System.out.println("Equal strings: " + isEqual);
    }

    public static boolean compareStrings(String str1, String str2) {
        if (str1.length() != str2.length()) {
            return false;
        }

        for (int i = 0; i < str1.length(); i++) {
            if (str1.charAt(i) != str2.charAt(i)) {
                return false;
            }
        }

        return true;
    }
}
```

```
Equal strings: false
```

3) Given string "<https://www.amazon.com/demo?test=abc>", return ["https","www","amazon","com","demo","test","abc"]

```
public class stringSplit {
    Run | Debug
    public static void main(String[] args) {
        String str = "https://www.amazon.com/demo?test=abc";

        String[] segments = str.split(regex:"[/?:=.]+" );
        for (String segment : segments) {
            System.out.println(segment);
        }
    }
}
```

```
https
www
amazon
com
demo
test
abc
```

4) Given a list of string array, combine them into one string sentence, return the string sentence

```
StringHw.java x
1 public class StringHw {
2
3     public static void main(String[] args) {
4         String s1 = "https://www.amazon.com/demo?test=abc";
5         String[] ans = s1.split(regex: "\\://|\\.|\\/|\\?|\\=");
6         for (int i = 0; i < ans.length; i++) {
7             System.out.println(ans[i]);
8         }
9     }
10 }
```

This is the first array And this is the second array Finally, the third array

Topic: final

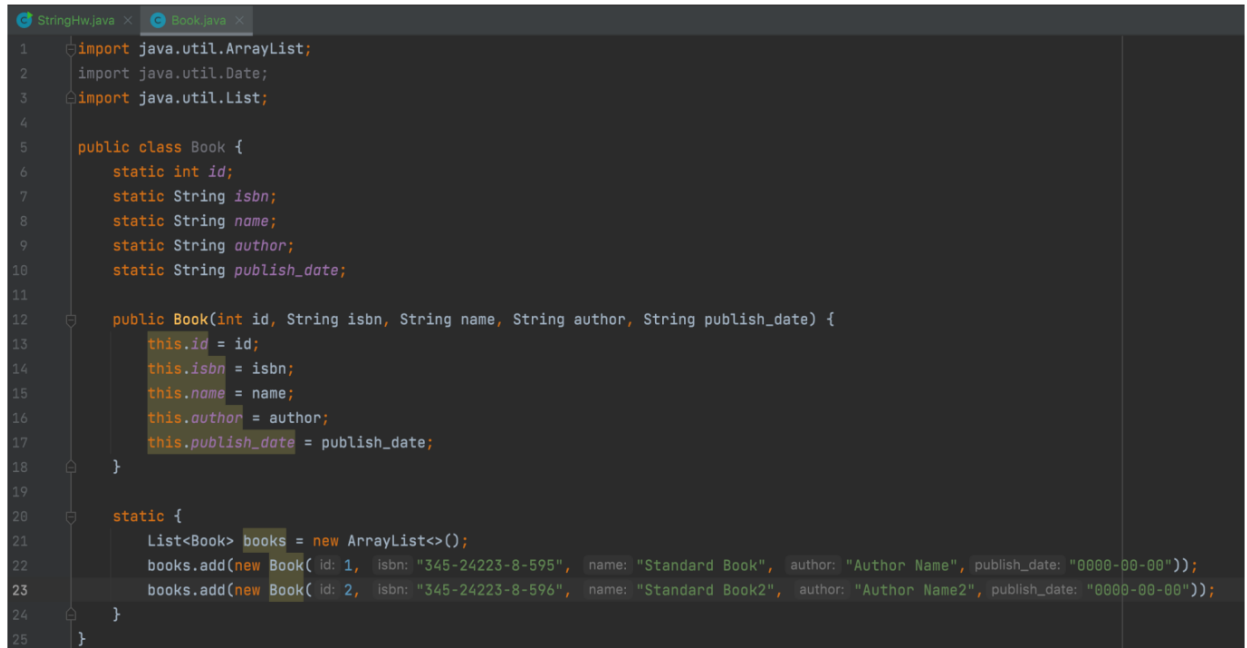
Problems: define a final class and final method and final variable, modify the value of the variable in final method

其实就是告诉大家改不了

Topic: Static

Problems:

- 1) Given a database table “Book” with columns (id, isbn, name, author, publish date), define a java class that matches this table and then use a static block to initialize this table with some records



```
1 import java.util.ArrayList;
2 import java.util.Date;
3 import java.util.List;
4
5 public class Book {
6     static int id;
7     static String isbn;
8     static String name;
9     static String author;
10    static String publish_date;
11
12    public Book(int id, String isbn, String name, String author, String publish_date) {
13        this.id = id;
14        this.isbn = isbn;
15        this.name = name;
16        this.author = author;
17        this.publish_date = publish_date;
18    }
19
20    static {
21        List<Book> books = new ArrayList<>();
22        books.add(new Book(1, "345-24223-8-595", "Standard Book", "Author Name", "0000-00-00"));
23        books.add(new Book(2, "345-24223-8-596", "Standard Book2", "Author Name2", "0000-00-00"));
24    }
25 }
```

- 2) Define a Java class “BookSeller” and then define a static inner class “Book”, and use a static method “sellBooks” to initialize several books, and in the main method display all the books by calling the “sellBooks” method

```
StringHw.java x Book.java x BookSeller.java x
3
4 public class BookSeller {
5     static class Book {
6         private int id;
7         private String isbn;
8         private String name;
9         private String author;
10        private String publish_date;
11
12        public Book(int id, String isbn, String name, String author, String publish_date) {
13            this.id = id;
14            this.isbn = isbn;
15            this.name = name;
16            this.author = author;
17            this.publish_date = publish_date;
18        }
19
20        @Override
21        public String toString() {
22            return "Book{" +
23                "isbn=" + isbn + '\'' +
24                "name=" + name + '\'' +
25                "author=" + author + '\'' +
26                "publish_date=" + publish_date + '\'' +
27                '}';
28        }
29    }
30    public static void main(String[] args) {
31        sellBooks();
32    }
33    public static void sellBooks() {
34        List<Book> books = new ArrayList<>();
35        books.add(new Book(1, "345-24223-8-595", "Standard Book", "Author Name", "0000-00-00"));
36        books.add(new Book(2, "345-24223-8-596", "Standard Book2", "Author Name2", "0000-00-00"));
37        System.out.println("Books available for sale:");
38        for (Book book : books) {
39            System.out.println(book.toString());
40        }
41    }
42 }
```

Topic: **OOP**

Problems:

- 1) Define an interface "DatabaseConnection" and then define class "OracleConnection", "MySQLConnection", "SqlServerConnection". They should all implements the "getConnection" method from the interface. The method should initialize data source and return a database connection.

```

interface DatabaseConnection {
    String getConnection();
}

class OracleConnection implements DatabaseConnection {
    @Override
    public String getConnection() {
        // Initialize data source and return Oracle database connection
        System.out.println(x:"Connected to Oracle database");
        // Your code for establishing Oracle database connection
        String databaseConnection = "Oracle";
        return databaseConnection;
    }
}

class MySqlConnection implements DatabaseConnection {
    @Override
    public String getConnection() {
        // Initialize data source and return MySQL database connection
        System.out.println(x:"Connected to MySQL database");
        // Your code for establishing MySQL database connection
        String databaseConnection = "MySQL";
        return databaseConnection;
    }
}

class SqlServerConnection implements DatabaseConnection {
    @Override
    public String getConnection() {
        // Initialize data source and return SQL Server database connection
        System.out.println(x:"Connected to SQL Server database");
        // Your code for establishing SQL Server database connection
        String databaseConnection = "SQL";
        return databaseConnection;
    }
}

```

```

public class DataBase {
    Run | Debug
    public static void main(String[] args) {
        DatabaseConnection oracleConnection = new OracleConnection();
        oracleConnection.getConnection();

        DatabaseConnection mySqlConnection = new MySqlConnection();
        mySqlConnection.getConnection();

        DatabaseConnection sqlServerConnection = new SqlServerConnection();
        sqlServerConnection.getConnection();
    }
}

```

```

Connected to Oracle database
Connected to MySQL database
Connected to SQL Server database

```

- 2) Define an abstract class "CreditCard" which contains fields (holderName, cardNumber, accountBalance, cardType), and not-implemented method "isCardAcceptable" with argument cardType, and implemented method "payBill(double bill)". Define two classes "VisaCard" and "MasterCard" both should inherit this "CreditCard" class and you should define constructor for both classes and implement the "isCardAcceptable" method.

```
abstract class CreditCard {
    private String holderName;
    private String cardNumber;
    private double accountBalance;
    private String cardType;

    public CreditCard(String holderName, String cardNumber, double accountBalance, String cardType) {
        this.holderName = holderName;
        this.cardNumber = cardNumber;
        this.accountBalance = accountBalance;
        this.cardType = cardType;
    }

    public abstract boolean isCardAcceptable(String cardType);

    public void payBill(double bill) {
        if (bill <= accountBalance) {
            accountBalance -= bill;
            System.out.println("Payment of $" + bill + " successfully processed.");
            System.out.println("Remaining account balance: $" + accountBalance);
        } else {
            System.out.println("Insufficient account balance to pay the bill.");
        }
    }
}
```

```
// Getters and setters (optional)
> public String getHolderName() { ...
>
> public void setHolderName(String holderName) { ...
>
> public String getCardNumber() { ...
>
> public void setCardNumber(String cardNumber) { ...
>
> public double getAccountBalance() { ...
>
> public void setAccountBalance(double accountBalance) { ...
>
> public String getCardType() { ...
>
> public void setCardType(String cardType) { ...
>
> }
```

```

class VisaCard extends CreditCard {
    public VisaCard(String holderName, String cardNumber, double accountBalance, String cardType) {
        super(holderName, cardNumber, accountBalance, cardType);
    }

    @Override
    public boolean isCardAcceptable(String cardType) {
        return cardType.equalsIgnoreCase(anotherString:"Visa");
    }
}

class MasterCard extends CreditCard {
    public MasterCard(String holderName, String cardNumber, double accountBalance, String cardType) {
        super(holderName, cardNumber, accountBalance, cardType);
    }

    @Override
    public boolean isCardAcceptable(String cardType) {
        return cardType.equalsIgnoreCase(anotherString:"MasterCard");
    }
}

```

3) Implement static and dynamic polymorphism.

Static:

```

1  class Calculator {
2      public static int add(int num1, int num2) {
3          return num1 + num2;
4      }
5
6      public static int add(int num1, int num2, int num3) {
7          return num1 + num2 + num3;
8      }
9  }
10
11 public class Static {
12     Run | Debug
13     public static void main(String[] args) {
14         int sum1 = Calculator.add(num1:5, num2:10);
15         System.out.println("Sum 1: " + sum1);
16
17         int sum2 = Calculator.add(num1:2, num2:4, num3:6);
18         System.out.println("Sum 2: " + sum2);
19     }
20 }

```

Dynamic:

```

class Shape {
    public void display() {
        System.out.println(x:"This is a shape.");
    }
}

class Circle extends Shape {
    @Override
    public void display() {
        System.out.println(x:"This is a cricle.");
    }
}

class Triangle extends Shape {
    @Override
    public void display() {
        System.out.println(x:"This is a triangle.");
    }
}

public class Dynamic {
    Run | Debug
    public static void main(String[] args) {
        Shape shape1 = new Shape();
        Shape circle = new Circle();
        Shape triangle = new Triangle();

        shape1.display();
        circle.display();
        triangle.display();
    }
}

```

```

This is a shape.
This is a cricle.
This is a triangle.

```

Topic: Design Pattern

Problems

- 1) Create a singleton class called "AppleDesignerFactory"

```
1 public class AppleDesignerFactory {
2     private static AppleDesignerFactory instance;
3     private AppleDesignerFactory() {}
4     public static AppleDesignerFactory getInstance() {
5         if (instance == null) {
6             instance = new AppleDesignerFactory();
7         }
8
9         return instance;
10    }
11 }
```

- 2) Create a factory pattern called "CurrencyExchange" which takes in the country name and return the currency object for that country.

```

interface Currency {
    String getCurrencyName();
}

class USD implements Currency {
    @Override
    public String getCurrencyName() {
        return "USD";
    }
}

class RMB implements Currency {
    @Override
    public String getCurrencyName() {
        return "RMB";
    }
}

class CurrencyExchange {
    public static Currency getCurrency(String country) {
        switch (country) {
            case "USA":
                return new USD();
            case "CHINA":
                return new RMB();
            default:
                throw new IllegalArgumentException(s:"No Currency supports this country.");
        }
    }
}

public class CurrencyCountry {
    Run | Debug
    public static void main(String[] args) {
        Currency usCurrency = CurrencyExchange.getCurrency(country:"USA");
        System.out.println("Currency name: " + usCurrency.getCurrencyName());

        Currency chinaCurrency = CurrencyExchange.getCurrency(country:"CHINA");
        System.out.println("Currency name: " + chinaCurrency.getCurrencyName());
    }
}

```

Currency name: USD
Currency name: RMB

- 3) Implement the (in-class) PARKING LOT OOP design system -> your implementation should include main method and is runnable. 这题其实大家可以直接看看网上的各种写法

```
1  import java.util.ArrayList;
2  import java.util.List;
3
4  class ParkingSpace {
5      private int id;
6      private Vehicle vehicle;
7
8      public ParkingSpace(int id) {
9          this.id = id;
10     }
11
12     public int getId() {
13         return id;
14     }
15
16     public boolean isOccupied() {
17         return vehicle != null;
18     }
19
20     public Vehicle getVehicle() {
21         return vehicle;
22     }
23
24     public void parkVehicle(Vehicle vehicle) {
25         this.vehicle = vehicle;
26     }
27
28     public void removeVehicle() {
29         this.vehicle = null;
30     }
31 }
```

```
class Vehicle {
    private String registrationNumber;

    public Vehicle(String registrationNumber) {
        this.registrationNumber = registrationNumber;
    }

    public String getRegistrationNumber() {
        return registrationNumber;
    }
}
```

```

class ParkingLot {
    private List<ParkingSpace> parkingSpaces;

    public ParkingLot(int capacity) {
        parkingSpaces = new ArrayList<>(capacity);
        for (int i = 0; i < capacity; i++) {
            parkingSpaces.add(new ParkingSpace(i + 1));
        }
    }

    public boolean parkVehicle(Vehicle vehicle) {
        for (ParkingSpace space : parkingSpaces) {
            if (!space.isOccupied()) {
                space.parkVehicle(vehicle);
                return true;
            }
        }
        return false;
    }

    public boolean removeVehicle(Vehicle vehicle) {
        for (ParkingSpace space : parkingSpaces) {
            if (space.isOccupied() && space.getVehicle().equals(vehicle)) {
                space.removeVehicle();
                return true;
            }
        }
        return false;
    }

    public void displayParkingStatus() {
        System.out.println("Parking Lot Status:");
        for (ParkingSpace space : parkingSpaces) {
            if (space.isOccupied()) {
                System.out.println("Space: " + space.getId() + ", Vehicle: " + space.getVehicle().getRegistrationNumber());
            } else {
                System.out.println("Space: " + space.getId() + ", Empty");
            }
        }
    }
}

```

```
public class Parking {  
    Run | Debug  
    public static void main(String[] args) {  
        // Create a parking lot with 3 spaces  
        ParkingLot parkingLot = new ParkingLot(capacity:3);  
  
        // Create vehicles  
        Vehicle vehicle1 = new Vehicle(registrationNumber:"A:43212");  
        Vehicle vehicle2 = new Vehicle(registrationNumber:"P:123567");  
  
        // Park vehicles  
        boolean vehicle1Parked = parkingLot.parkVehicle(vehicle1);  
        boolean vehicle2Parked = parkingLot.parkVehicle(vehicle2);  
  
        if (vehicle1Parked && vehicle2Parked) {  
            System.out.println(x:"Vehicles parked successfully.");  
        } else {  
            System.out.println(x:"Failed to park vehicles.");  
        }  
  
        // Display parking lot status  
        parkingLot.displayParkingStatus();  
  
        // Remove vehicle  
        boolean vehicleRemoved = parkingLot.removeVehicle(vehicle1);  
        if (vehicleRemoved) {  
            System.out.println(x:"Vehicle removed successfully.");  
        } else {  
            System.out.println(x:"Failed to remove vehicle.");  
        }  
  
        // Display parking lot status again  
        parkingLot.displayParkingStatus();  
    }  
}
```

```
Vehicles parked successfully.
Parking Lot Status:
Space: 1, Vehicle: A:43212
Space: 2, Vehicle: P:123567
Space: 3, Empty
Vehicle removed successfully.
Parking Lot Status:
Space: 1, Empty
Space: 2, Vehicle: P:123567
Space: 3, Empty
```

Topic: Collection

Problems:

- 1) (Set)Find true friends: Given two arraylists containing friend names, find the true friends that appear in both list.

```
import java.util.ArrayList;

public class TrueFriend {
    Run | Debug
    public static void main(String[] args) {
        ArrayList<String> list1 = new ArrayList<>();
        list1.add(e:"James");
        list1.add(e:"Wade");
        list1.add(e:"Bob");
        list1.add(e:"Alice");

        ArrayList<String> list2 = new ArrayList<>();
        list2.add(e:"Bob");
        list2.add(e:"Alice");
        list2.add(e:"Adam");
        list2.add(e:"Joseph");

        list1.retainAll(list2);

        System.out.println("True friends: " + list1);
    }
}
```

Use Set:

```

import java.util.ArrayList;
import java.util.HashSet;
import java.util.Set;

public class TrueFriend {
    Run | Debug
    public static void main(String[] args) {
        ArrayList<String> list1 = new ArrayList<>();
        list1.add(e:"James");
        list1.add(e:"Wade");
        list1.add(e:"Bob");
        list1.add(e:"Alice");

        ArrayList<String> list2 = new ArrayList<>();
        list2.add(e:"Bob");
        list2.add(e:"Alice");
        list2.add(e:"Adam");
        list2.add(e:"Joseph");

        Set<String> set1 = new HashSet<>(list1);
        Set<String> set2 = new HashSet<>(list2);

        set1.retainAll(set2);

        System.out.println("True friends: " + set1);
    }
}

```

- 2) (Map) Given a string, output duplicate characters and their counts

```

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

public class Duplicate {
    Run | Debug
    public static void main(String[] args) {
        String input = "The train does not leave at 12 AM.";

        Map<Character, Integer> charCounts = new HashMap<>();

        for (char c : input.toCharArray()) {
            if (!Character.isLetter(c) || Character.isWhitespace(c)) {
                continue;
            }
            char lowercaseChar = Character.toLowerCase(c);
            charCounts.put(lowercaseChar, charCounts.getOrDefault(lowercaseChar, defaultValue:0) + 1);
        }

        for (Map.Entry<Character, Integer> entry : charCounts.entrySet()) {
            if (entry.getValue() > 1) {
                System.out.println("Character: " + entry.getKey() + ", Count: " + entry.getValue());
            }
        }
    }
}

```

```
Character: a, Count: 4  
Character: e, Count: 4  
Character: n, Count: 2  
Character: o, Count: 2  
Character: t, Count: 4
```

- 3) Use a map to simulate database table, key should be the primary key (assume only one column), value is the record, your simulation should include CRUD operation methods


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

public class DatabaseTableSimulation {
    private Map<Integer, String> table;

    public DatabaseTableSimulation() {
        table = new HashMap<>();
    }

    public void createRecord(int primaryKey, String record) {
        if (!table.containsKey(primaryKey)) {
            table.put(primaryKey, record);
            System.out.println(x:"Record created successfully.");
        } else {
            System.out.println(x:"Record with primary key already exists.");
        }
    }

    public void readRecord(int primaryKey) {
        String record = table.get(primaryKey);
        if (record != null) {
            System.out.println("Record found: " + record);
        } else {
            System.out.println(x:"Record not found.");
        }
    }

    public void updateRecord(int primaryKey, String updatedRecord) {
        if (table.containsKey(primaryKey)) {
            table.put(primaryKey, updatedRecord);
            System.out.println(x:"Record updated successfully.");
        } else {
            System.out.println(x:"Record not found.");
        }
    }

    public void deleteRecord(int primaryKey) {
        if (table.containsKey(primaryKey)) {
            table.remove(primaryKey);
            System.out.println(x:"Record deleted successfully.");
        } else {
            System.out.println(x:"Record not found.");
        }
    }
}
```

```

public static void main(String[] args) {
    DatabaseTableSimulation tableSimulation = new DatabaseTableSimulation();

    // Create records
    tableSimulation.createRecord(primaryKey:1, record:"Record 1");
    tableSimulation.createRecord(primaryKey:2, record:"Record 2");
    tableSimulation.createRecord(primaryKey:3, record:"Record 3");

    // Read records
    tableSimulation.readRecord(primaryKey:1);
    tableSimulation.readRecord(primaryKey:4);

    // Update records
    tableSimulation.updateRecord(primaryKey:3, updatedRecord:"Updated Record 3");
    tableSimulation.updateRecord(primaryKey:4, updatedRecord:"Updated Record 4");

    // Delete records
    tableSimulation.deleteRecord(primaryKey:1);
    tableSimulation.deleteRecord(primaryKey:3);
    // Check Again
    tableSimulation.readRecord(primaryKey:1);
    tableSimulation.readRecord(primaryKey:3);
}

```

```

Record created successfully.
Record created successfully.
Record created successfully.
Record found: Record 1
Record not found.
Record updated successfully.
Record not found.
Record deleted successfully.
Record deleted successfully.
Record not found.
Record not found.

```