

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Sichtbarkeits- und Zugriffsmodifizierer wie public, usw. dürfen nicht weggelassen werden.
- In allen Klassen dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen, außer es handelt sich um reine Ausgabe- oder Eingabe Methoden.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

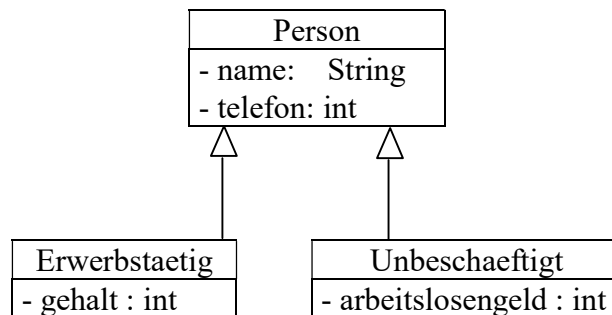
AUFGABEN

ACHTUNG: Die Klassenarbeit besteht aus genau einer Aufgabe (die aus Teilaufgaben besteht). Die gesamte Aufgabe muß als genau ein Programm in einem Projekt implementiert werden.

I)

50P

Gegeben ist das folgende abgespeckte UML-Diagramm:



1)

32P

Erzeugen Sie aus dem folgenden UML-Diagramm die entsprechenden Klassen mit den jeweiligen Konstruktoren, set- und get-Methoden.

Die Konstruktoren müssen jeweils Parameter enthalten.

2)

9P

Erzeugen Sie in der Methode main (jeweils durch eine Anweisung)

a) den Erwerbstätigen "Schaffer" mit der Telefonnummer 4711, dem Gehalt von 1500 Euro

b) den Unbeschäftigten "Schröder" mit der Telefonnummer 4712 und Arbeitslosengeld von 400 Euro

c) die Person "Schulze" mit der Telefonnummer 12345

d)

Herr Schaffer bekommt 100 Euro mehr Gehalt.

Realisieren Sie dies durch genau eine einzige Anweisung (keine neue Methode implementieren), die zum alten Gehalt die 100 Euro dazu addiert.

In der Anweisung muss also der alte Gehalt ermittelt werden.

3)

9P

a) 6P

Erzeugen Sie zusätzlich in der Klasse "Erwerbstaetig" die Methode

... vergleiche(...)

Diese soll von 2 Erwerbstaetigen den Erwerbstaetigen zurückliefern, der das größere Gehalt hat. Bei 2 Erwerbstaetigen mit dem gleichen Gehalt ist es egal, wer von den beiden zurückgeliefert wird.

b) 3P

Vergleichen Sie in main mit Hilfe der Methode ... vergleiche(...) den Erwerbstaetigen "Schaffer" mit sich selbst und speichern Sie das Ergebnis in einer entsprechenden Variablen ab.

Lösung:

1)

```
public class Main_E2FI_8_4_08_nr1 {
    public static void main(String[] args){
        Erwerbstaetig e = new Erwerbstaetig("Schaffer", 4711, 1500);    // 2P
        Unbeschaeftigt u = new Unbeschaeftigt("Schroeder", 4712, 400); // 2P
        Person p = new Person("Schulze",12345);                        // 2P
        e.setGehalt(e.getGehalt()+100);                                // 3P
        Erwerbstaetig temp = e.vergleiche(e);                          // 3P
    }
}

class Person{                                                         // 12P
    private String name;      // 1P
    private int telefon;      // 1P

    public Person (String pName, int pTelefon){ // 2P
        name = pName;
        telefon = pTelefon;
    }

    public void setTelefon(int pTelefon){ //2P
        telefon = pTelefon;
    }

    public void setName(String pName){ // 2P
        name = pName;
    }

    public int getTelefon(){ // 2P
        return(telefon);
    }

    public String getName(){ // 2P
        return(name);
    }
}

class Unbeschaeftigt extends Person{ //1P                               // 10P
    private int arbeitlosengeld;    // 1P

    public Unbeschaeftigt(String pName, int pTelefon, // 4P
        int pArbeitslosengeld){
        super(pName, pTelefon);
        arbeitlosengeld = pArbeitslosengeld;
    }

    public void setArbeitslosengeld(int pArbeitslosengeld){ // 2P
        arbeitlosengeld = pArbeitslosengeld;
    }

    public int getArbeitslosengeld(){ // 2P
        return(arbeitslosengeld);
    }
}
```

```
class Erwerbstaetig extends Person{ // 1P
    private int gehalt ; // 1P

    public Erwerbstaetig(String pName, int pTelefon, int pGehalt){ // 4P
        super(pName, pTelefon);
        gehalt = pGehalt;
    }

    public void setGehalt(int pGehalt){ // 2P
        gehalt = pGehalt;
    }

    public int getGehalt(){ // 2P
        return(gehalt);
    }

    public Erwerbstaetig vergleiche(Erwerbstaetig pErwerbstaetig){ // 6P
        if(this.gehalt < pErwerbstaetig.gehalt){
            return pErwerbstaetig;
        }
        else{
            return this;
        }
    }
}
```

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Sichtbarkeits- und Zugriffsmodifizierer wie public, usw. dürfen nicht weggelassen werden.
- In allen Klassen dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen, außer es handelt sich um reine Ausgabe- oder Eingabe Methoden.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punktabzug !!!!

AUFGABEN

I) 51P

Die Verwaltung eines Bauernhofs soll mit Hilfe der OOP auf EDV umgestellt werden.
In einem Gespräch zwischen der EDV-Firma „Makall“ und den Besitzern des Bauernhofs wurde folgendes festgehalten:

Der Bauernhof gehört genau 2 Besitzern d.h. Personen und besteht aus genau einem Bauernhaus und einem Kuhstall, in dem maximal 10 Kühe untergebracht werden können..
Die Klasse Kuh wurde schon implementiert (mit den Attributen name und alter und den entsprechenden Methoden).

Aus Gründen der Vereinfachung besitzen die Klassen "Besitzer", "Person" und "Bauernhaus" jeweils genau ein Attribut

1) 16P
Erstellen Sie ein UML-Diagramm (mit den entsprechenden Attributen, aber ohne Methoden), das diesen Fall modelliert.

2) 7P
Implementieren Sie die Klasse "Person" mit den nötigen Attributen und Methoden.

3) 9P
Implementieren Sie die Klasse "Besitzer" mit den nötigen Attributen und Methoden.

4) 4P
a)
Erzeugen Sie die Klasse "Bauernhof" mit allen nötigen Attributen.

b) 5P
Erzeugen Sie genau einen Konstruktor (mit genau 4 Parametern). Falls ein Array benötigt wird, soll dieses nicht dynamisch sein und die Länge 10 haben.

c)

10P

Erstellen Sie in der Klasse Bauernhof eine Methode

... anfüegenKuh (Kuh pKuh) :

fügt eine Kuh an das Ende der bisherigen gespeicherten Kühe an.

Wenn das Feld voll ist, werden keine neuen Kühe angefügt.

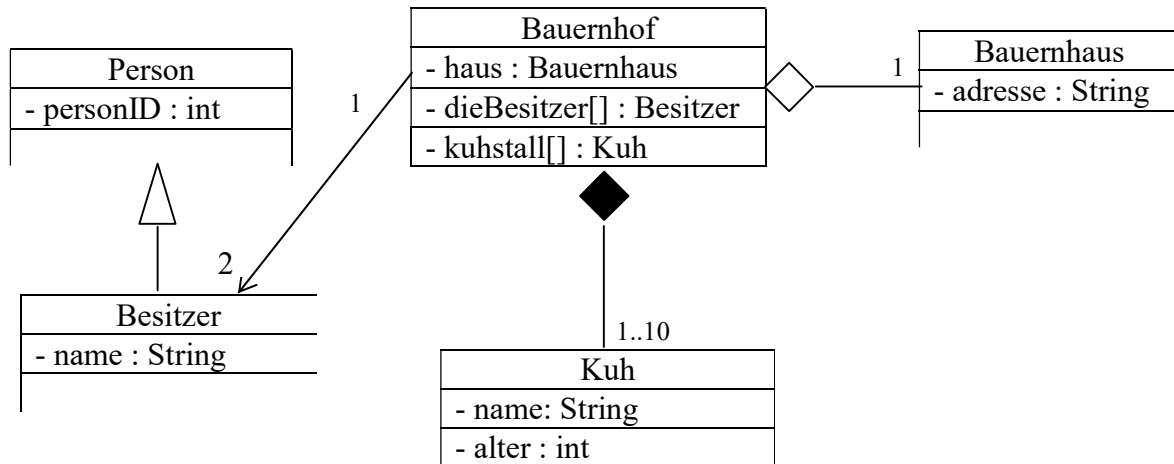
Bemerkung:

In allen Klassen dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen, außer es handelt sich um reine Ausgabe-Methoden.

Alle Teilaufgaben - sofern es sich um Programmieraufgaben handelt - müssen in **einem** Programm realisiert werden

Lösung:

1)



2)

7P

```
class Person{
    private int personID;

    public Person(int personID) {
        this.personID = personID;
    }

    public int getPersonID() {
        return personID;
    }

    public void setPersonID(int personID) {
        this.personID = personID;
    }
}
```

3)

9P

```
class Besitzer extends Person {
    private String name;

    public Besitzer(int personID, String pName){
        super(personID);
        name=pName;
    }

    public String getName() {
        return name;
    }

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

4)

```
class Bauernhof{
    private String adresse;
    private Kuh[] kuhstall;
    private Besitzer[] dieBesitzer;
    private Bauernhaus haus;

    public Bauernhof(String adresse, Besitzer b1, Besitzer b2,
                      Bauernhaus haus){
        this.adresse = adresse;
        dieBesitzer=new Besitzer[2];
        dieBesitzer[0]=b1;
        dieBesitzer[1]=b2;
        this.haus=haus;
        kuhstall = new Kuh[10];
    }

    public void an fuegenKuh(Kuh k){
        for(int i=0;i<10;i++){
            if(kuhstall[i]==null){
                kuhstall[i]=k;
                return;
            }
        }
    }

    public void printKuhstall(){
        for(int i=0;i<10;i++){
            if(kuhstall[i]!=null){
                sout ("Kuhname="+kuhstall[i].getName());
                sout ("Kuhgewicht="+kuhstall[i].getGewicht());
            }
        }
    }
}
```


Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Sichtbarkeits- und Zugriffsmodifizierer wie public, usw. dürfen nicht weggelassen werden.
- In allen Klassen dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen, außer es handelt sich um reine Ausgabe- oder Eingabe Methoden.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN

ACHTUNG: Die Klassenarbeit besteht aus genau einer Aufgabe (die aus Teilaufgaben besteht). Die gesamte Aufgabe muß als genau ein Programm in einem Projekt implementiert werden.

I) 50P

Zur Information:

Eine Teilstruktur einer Firma soll modelliert werden:

Eine Abteilung hat einen Namen (z.B. Elektroabteilung), besitzt einen Abteilungsleiter (z.B. die Person mit Namen Lowski), und besteht aus mehreren Personen.

Eine Person hat eine Identitätsnummer idNr und einen Namen.

1) 15P

Erstellen Sie die Klasse Person (mit genau den 2 Attributen idNr und name, genau 2 set-Methoden, genau 2 get-Methoden und genau einem Konstruktor mit 2 Parametern) Zusätzlich muß noch genau die Methode printAllAttributes(...) erzeugt werden, die die Werte aller Attribute dieser Klasse auf dem Bildschirm ausgibt.

2) 24P

a) Erstellen Sie die Klasse Abteilung (mit genau den 3 Attributen "name", der Person "leiter" und dem Feld "diePersonen" und die zu name und leiter zugehörigen get - und set-Methoden und genau einem Konstruktor (mit genau 3 Parametern, wobei ein Parameter die Anzahl der Personen in der Abteilung festlegt).

b) Zusätzlich sollen genau noch folgende Methoden erstellt werden:

public void setPerson(Person pPerson, int index)
speichert eine Person an einer bestimmten Stelle des Feldes.

public Person getPerson(int index)
gibt die Person an einer bestimmten Stelle des Feldes zurück.

public void printAllAttributes()
Gibt die Attribute name, leiter und alle Personen der Abteilung auf dem Bildschirm aus.

3) 12P

Erstellen sie eine Abteilung (der Variablennamen soll dieAbteilung heißen) mit dem Namen "Elektroabteilung", deren Leiter "Lowski" (Identitätsnummer 10) und den 2 Personen "Maier" mit Identitätsnummer 20 und "Maurer" mit Identitätsnummer 30.

Geben Sie mit genau einem einzigen Aufruf (einer bestimmten Methode) die Personen (mit Idenditätsnummer und Name) auf dem Bildschirm aus.

Lösung:

```
public class Startklasse {
    public static void main(String[] args) {
        Abteilung dieAbteilung = new Abteilung("Elektro", // 3P
            new Person(10,"Lowski"), 2);
        dieAbteilung.setPerson(new Person(20,"Maier"),0); // 3P
        dieAbteilung.setPerson(new Person(30,"Maurer"),1); // 3P
        dieAbteilung.printAllAttributes(); // 2P
    }
}

class Person {
    private int idNr; // 1P
    private String name; // 1P

    public Person(int idNr, String name) { // 2P
        this.idNr = idNr;
        this.name = name;
    }

    public int getIdNr() { // 2P
        return idNr;
    }

    public void setIdNr(int idNr) { // 2P
        this.idNr = idNr;
    }

    public String getName() { // 2P
        return name;
    }

    public void setName(String name) { // 2P
        this.name = name;
    }

    public void printAllAttributes(){ // 3P
        System.out.println("Personen-Id= " + idNr);
        System.out.println("Personen-Id= " + name);
    }
}
```

```

class Abteilung {
    private String name; // 1P
    private Person leiter; // 1P
    private Person[] ihrePersonen; // 1P

    public Abteilung(String abteilung, Person leiter, int anzahl){ //3P
        this.name = abteilung;
        this.leiter = leiter;
        ihrePersonen = new Person[anzahl];
    }

    public void setName(String pName) { // 2P
        name = pName;
    }

    public String getName() { // 2P
        return (name);
    }

    public void setLeiter(Person pName) { // 2P
        leiter = pName;
    }

    public Person getLeiter() { // 2P
        return (leiter);
    }

    public void setPerson(Person pName, int index) { // 3P
        ihrePersonen[index] = pName;
    }

    public Person getPerson(int index) { // 3P
        return (ihrePersonen[index]);
    }

    public void printAllAttributes(){ // 4P
        System.out.println("Abteilungsname= " + name);
        System.out.println("Leiter= " + leiter);
        for(int i=0; i<ihrePersonen.length;i++) {
            ihrePersonen[i].printAllAttributes();
        }
    }
}

```

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Sichtbarkeits- und Zugriffsmodifizierer wie public, usw. dürfen nicht weggelassen werden.
- In allen Klassen dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen, außer es handelt sich um reine Ausgabe- oder Eingabe Methoden.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punktabzug !!!!

AUFGABEN

ACHTUNG: Die Klassenarbeit besteht aus genau einer Aufgabe (die aus Teilaufgaben besteht). Die gesamte Aufgabe muß als genau ein Programm in einem Projekt implementiert werden.

I)

54P

Zur Information:

Eine Teilstruktur eines Fußballvereins soll modelliert werden:

Ein Spieler ist eine Person (mit einem Namen) und kann i.A. verschiedene Rollen spielen (Torwart, Verteidiger, Mittelfeld, Stürmer: abgekürzt mit 'T', 'V', 'M', 'S').

Die für den Fußballverein wichtigen Eigenschaften eines Spielers sind (neben den Rollen, die er spielen kann) genau die Folgenden:

Die Zeit "zeit100" für einen 100 Meter-Lauf (in Sekunden) und die durchschnittliche Laufleistung "laufleistung", die er während eines Spiels vollbringt (in Kilometer).

1) 7P

Erstellen Sie die Klasse "Person" (mit genau dem Attributen "name", genau 1 set-Methode, genau 1 get-Methode und genau einem Konstruktor mit 1 Parameter)

2) 14P

Erstellen Sie die Klasse "Rolle" (mit genau den 2 Attributen "positionen", und "bewertungen").

Der Inhalt des Attributs "positionen" besteht aus den Jobs, die ein Spieler spielen kann.

Ein Allroundspieler kann z.B. maximal alle Jobs 'T', 'V', 'M', 'S' machen, während ein reiner Torwart nur den Job 'T' machen kann.

Der Inhalt des Attributs "bewertungen" besteht aus den Noten (von 1 bis 6 je einschließlich), die ein Spieler für jeden seiner Jobs - die er machen kann - bekommt. Zusätzlich müssen noch die zugehörigen get - und set-Methoden und genau ein Konstruktor (mit genau 2 Parametern) erzeugt werden

3) 26P

a)

Erstellen Sie die Klasse "Spieler" (mit genau den 2 Attributen "zeit100" und "laufleistung", genau 2 set-Methoden, genau 2 get-Methoden und genau einem Konstruktor mit 3 Parametern)

b)

Ein Spieler kann in einem Spiel eine Rolle (im obigen Sinn) spielen

Vervollständigen Sie die Klasse Spieler durch genau das Attribut "rolle" mit Datentyp "Rolle", so daß dies berücksichtigt wird.

Erzeugen Sie zusätzlich noch die dazugehörige get- und set- Methode.

c)

Zusätzlich muß noch genau die Methode printAllAtrubutes(...) erzeugt werden, die sämtliche Eigenschaften eines Spielers auf dem Bildschirm ausgibt:

den Namen "name" des Spielers,

die Zeit "zeit100" für einen 100 Meter-Lauf (in Sekunden),

die durchschnittliche Laufleistung "laufleistung" während eines Spiels (in Kilometer),

die verschiedene Rollen, die er spielen kann,

die Noten, die er für diese verschiedene Rollen bekommt.

4) 7P

Erstellen sie einen Spieler mit Namen "Gomez", der 100 Meter in 10 Sekunden läuft und während eines Spiels durchschnittlich 23 Kilometer rennt, Mittelfeld und Stürmer spielen kann und jeweils dafür mit der Note 2 bewertet wird.

Lösung:

```
public class Startklasse {  
    public static void main(String[] args) {  
        Rolle rolle1=new Rolle(new char[] {'M','S'}, new int[] {2,2}); // 3P  
        Spieler s1=new Spieler("Gomez", 10, 23 ); // 2P  
        s1.setRolle(rolle1); // 1P  
        s1.printAllAttributes(); // 1P  
    }  
}  
  
// 7P  
class Person {  
    private String name; // 1P  
  
    public Person(String name) { // 2P  
        this.name = name;  
    }  
  
    public String getName() { // 2P  
        return name;  
    }  
  
    public void setName(String name) { // 2P  
        this.name = name;  
    }  
}  
  
// 14P  
class Rolle{  
    private char[] positionen; // 1P  
    private int[] bewertungen; // 1P  
  
    public Rolle(char[] positionen, int[] bewertungen) { // 4P  
        this.positionen=positionen;  
        this.bewertungen=bewertungen;  
    }  
  
    public char[] getPositionen() { // 2P  
        return positionen;  
    }  
  
    public void setPositionen(char[] positionen) { // 2P  
        this.positionen = positionen;  
    }  
  
    public int[] getBewertungen() { // 2P  
        return bewertungen;  
    }  
  
    public void setBewertungen(int[] bewertungen) { // 2P  
        this.bewertungen = bewertungen;  
    }  
}
```

```

// 26P
class Spieler extends Person{                                // 1P
    private double zeit100;                                    // 1P
    private double laufleistung;                              // 1P
    private Rolle rolle;                                      // 1P

    public Spieler(String name, double zeit100, double ll){    // 4P
        super(name);
        this.zeit100=zeit100;
        this.laufleistung=ll;
    }

    public double getZeit100() {                                // 2P
        return zeit100;
    }

    public void setZeit100(double zeit100) {                   // 2P
        this.zeit100 = zeit100;
    }

    public double getLaufleistung() {                           // 2P
        return laufleistung;
    }

    public void setLaufleistung(double laufleistung) {         // 2P
        this.laufleistung = laufleistung;
    }

    public Rolle getRolle() {                                   // 2P
        return rolle;
    }

    public void setRolle(Rolle rolle) {                         // 2P
        this.rolle = rolle;
    }

    public void printAllAttributes(){                           // 6P
        int i;
        System.out.println("Name=" + getName());
        System.out.println("100-Meter-Zeit=" + getZeit100());
        System.out.println("Kilometer pro Spiele=" + getLaufleistung());
        System.out.print("spielt=");
        for(i=0;i< rolle.getPositionen().length;i++){
            System.out.print(rolle.getPositionen()[i]+"");
        }
        System.out.println();
        System.out.print("wird bewertet mit=");
        for(i=0;i< rolle.getBewertungen().length;i++){
            System.out.print(rolle.getBewertungen()[i]+"");
        }
    }
}

```