

	Familienname:		
	Matrikelnummer:	Kennzahl:	
	Lehrveranstaltung: 384.061: Objektorientiertes Programmieren		
Schriftliche Prüfung		Datum: 26. Jan. 2011 Ort: Audi Max	
			Schriftliche Note:
			Negative Note: die erforderliche Anzahl von _____ Punkten wurde nicht erreicht.

Beispiel 1 (20 Punkte)

Gesamtpunktzahl: 30

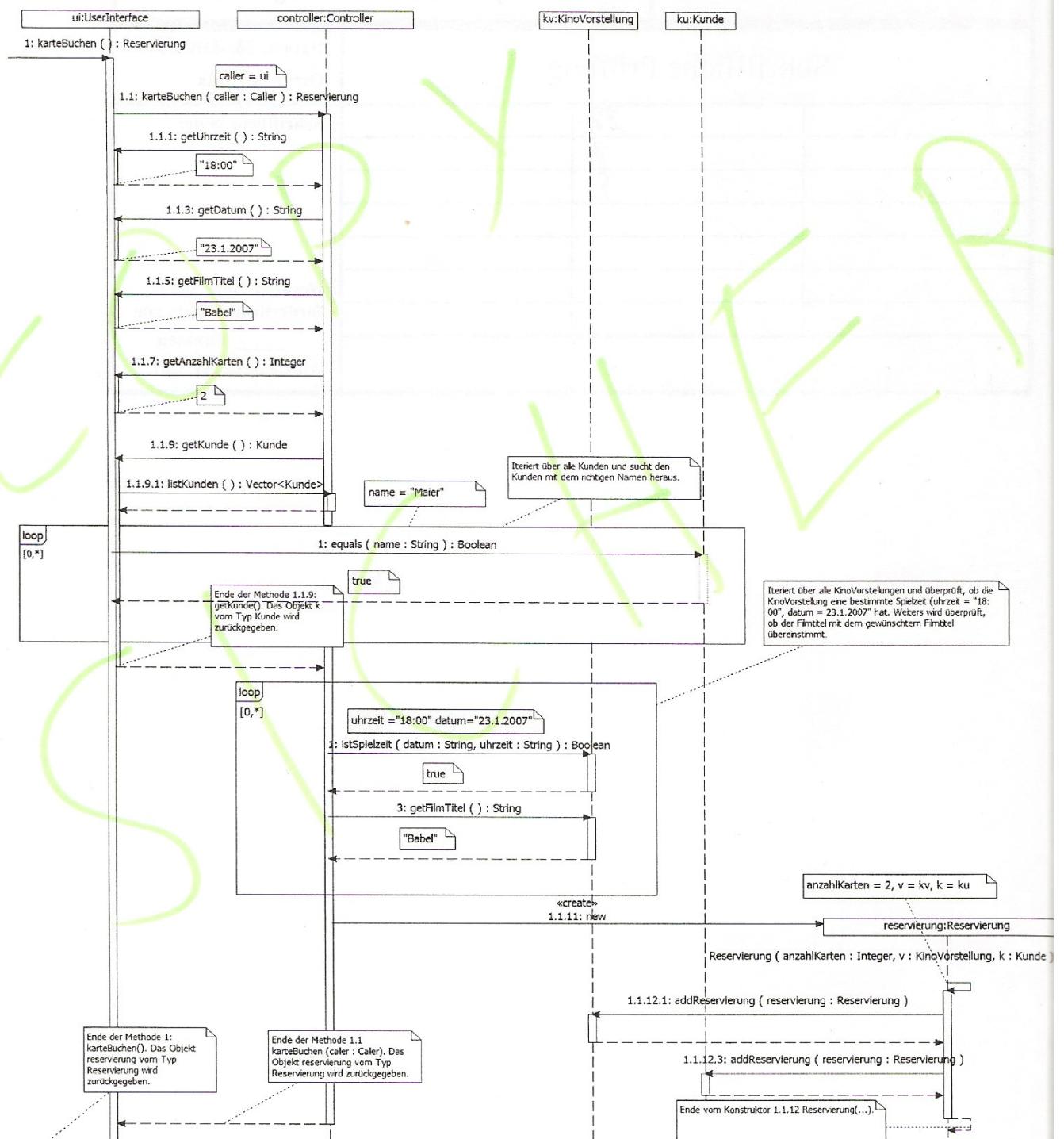
Beispiel 1a) (10 Punkte)

Gegeben sind ein Klassen- und ein Sequenzdiagramm eines Software-Entwurfs. Basierend auf diesen Diagrammen ist folgende Aufgabe zu lösen:

Implementieren Sie die Klassen (Methoden und Attribute) entsprechend dem Sequenz-, dem Klassendiagramm und dem angegebenen Code der Klasse **Angabe** (das Programm wird in der Klasse **Angabe** initialisiert).

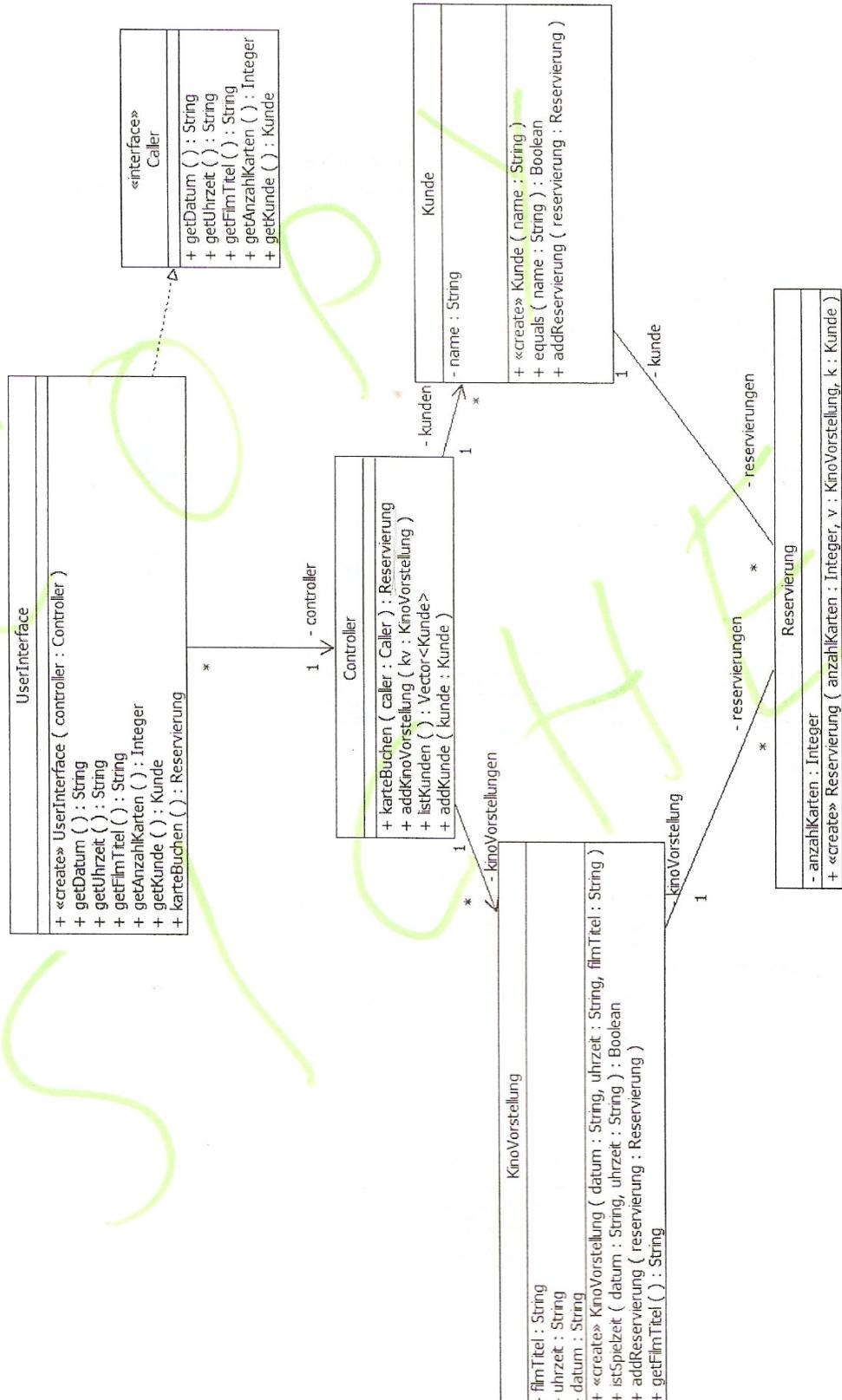
Das Sequenzdiagramm beschreibt den Buchungsvorgang einer Kinokarte. Eingaben eines Benutzers werden in der Klasse **UserInterface** „simuliert“, d.h. es reicht, wenn z.B. die Methode **UserInterface.getUhrzeit()** den String „18:00“ liefert – Sie brauchen daher keine interaktive Benutzereingabe zu implementieren. Die Klasse **Controller** stellt das Buchungssystem des Kinos dar.

Sequenzdiagramm:



Klassendiagramm

Assoziationen mit einem Pfeil (gerichtete Assoziationen) im Klassendiagramm bedeuten, dass jene Klasse beim Pfeilursprung eine Referenz (gespeichert in einer Instanzvariable) zur Klasse am Pfeilende besitzt. Die Referenzen sind den Assoziationen zugeordnet – z.B. Assoziation "Reservierung – Kunde": Ein Objekt von Typ Reservierung speichert die Referenz zu einem Objekt vom Typ Kunde in der Variable kunde. Ein Objekt von Typ Kunde speichert Referenzen zu Objekten vom Typ Reservierung im Vektor reservierungen. Das Minus „-“ vor den Bezeichnern der Variablen weist auf die Sichtbarkeit hin – in diesem Fall **private**.



Klassen

```
public class Angabe {  
  
    public static void main( String[] args ) {  
  
        Controller controller = new Controller();  
        UserInterface ui = new UserInterface( controller );  
        controller.addKinoVorstellung( new KinoVorstellung( "18:00", "23.1.2007", "Babel" ) );  
        controller.addKunde( new Kunde( "Maier" ) );  
  
        Reservierung reservierung = ui.karteBuchen(); // siehe oben links im Sequenzdiagramm  
    }  
}
```

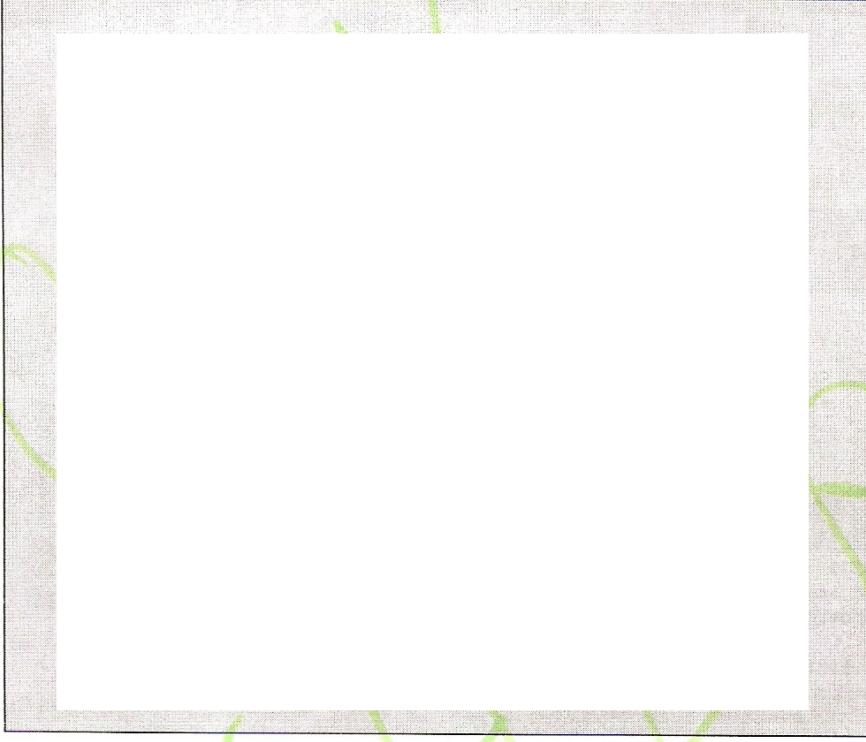
Vervollständigen Sie die Klassen:

Und schreiben Sie in alle mit  gekennzeichneten Felder den fehlenden Code!

```
-----  
public class Caller {  
    public String getUhrzeit();  
    public int getAnzahlKarten();  
    public String getFilmTitel();  
    public String getDatum();  
    public Kunde getKunde();  
}  
-----  
  
public class KinoVorstellung {  
    private String filmTitel;  
    private String datum;  
    private String uhrzeit;  
  
  
    public KinoVorstellung( String datum, String uhrzeit, String filmTitel){  
        this.filmTitel = filmTitel;  
        this.datum = datum;  
        this.uhrzeit = uhrzeit;  
    }  
  
    public boolean istSpielzeit(String datum, String uhrzeit){  
  
    }  
  
    public String getFilmTitel(){  
        return this.filmTitel;  
    }  
  
    public void addReservierung(Reservierung reservierung){  
  
    }  
}
```

Vervollständigen Sie die Klassen:

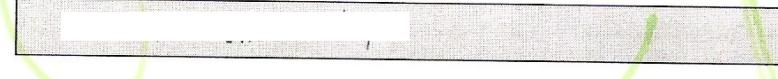
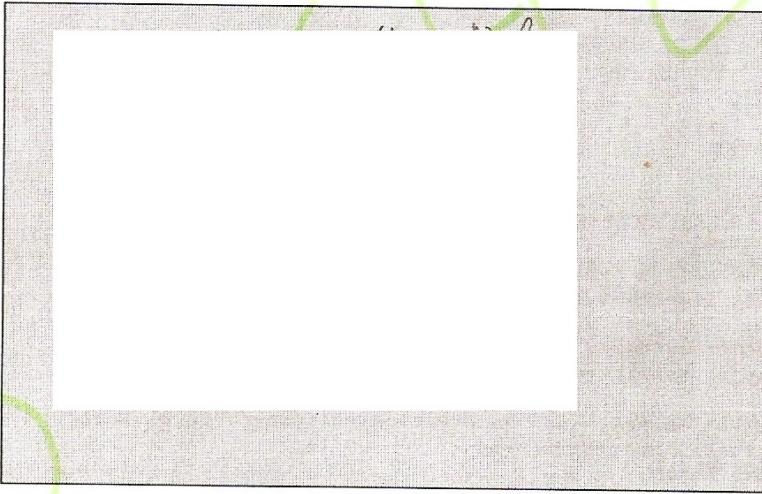
```
//-----  
  
public class Controller {  
  
    private Vector<KinoVorstellung> kinoVorstellungen=new Vector<KinoVorstellung>();  
    private Vector<Kunde> kunden=new Vector<Kunde>();  
  
    public Reservierung karteBuchen(Caller caller){  
  
    }  
  
    public Vector<Kunde> listKunden(){  
        return this.kunden;  
    }  
  
    public void addKunde(Kunde kunde){  
        this.kunden.add(kunde);  
    }  
  
    public void addKinoVorstellung(KinoVorstellung kv){  
        this.kinoVorstellungen.add(kv);  
    }  
}
```



Vervollständigen Sie die Klassen:

```
//-----  
public class Kunde {  
    private String name;  
  
    public Kunde(String name){  
        this.name=name;  
    }  
  
    public boolean equals(String name){  
        return this.name.equalsIgnoreCase(name);  
    }  
  
    public void addReservierung(Reservierung reservierung){  
        //-----  
    }  
}  
//-----  
public class Reservierung {  
    private Kunde kunde;  
    private int anzahlKarten;  
    private KinoVorstellung kinoVorstellung;  
  
    public Reservierung(int anzahlKarten,KinoVorstellung v, Kunde k){  
        this.kunde = k;  
        this.kinoVorstellung = v;  
        this.anzahlKarten = anzahlKarten;  
    }  
}
```

Vervollständigen Sie die Klassen:

```
//-----  
  
public class UserInterface implements Caller {  
  
    private Controller controller;  
    public UserInterface(Controller controller){  
        this.controller=controller;  
    }  
  
    public Reservierung karteBuchen(){  
        return controller.karteBuchen(this);  
    }  
  
    public String getUhrzeit(){  
          
    }  
  
    public int getAnzahlKarten(){  
        return 2;  
    }  
  
    public String getDatum(){  
          
    }  
  
    public String getFilmTitel(){  
          
    }  
  
    public Kunde getKunde(){  
          
    }  
}
```

Beispiel 1b) (10 Punkte)

Gegeben sind Implementierungen von sieben Klassen einer Autoreservierung (UI, ReservationCtrl, Car, Van, Sedan, Customer und Reservation).

Aufgabe: Zeichnen Sie das dazugehörige OOD-Sequenzdiagramm. Hinweis: Der erste Methoden-Aufruf („Startpunkt“) ist in der Methode UI.run(). Zeichnen Sie davon ausgehend alle Methodenaufrufe bis zum letztmöglichen Methodenauftrag (Programmende).

Wichtig: Geben Sie im Sequenzdiagramm alle Klassen- und Methodennamen an. Sie dürfen die Angabe von Typen vernachlässigen.

```
public class UI {
    public String getName() {
        return "Müller";
    }

    public String getLicenceNumber() {
        return "FS 12345678";
    }

    public void run(ReservationCtrl resCtrl) {
        Car car = resCtrl.getAvailableCar("Van");
        if (car == null) {
            System.out.println("No available car found");
            return;
        } else {
            Reservation res = resCtrl.reserveCar(car);
            System.out.println("Reservierung: " + res);
        }
    }
}

public class ReservationCtrl {
    private java.util.Vector<Car> cars;
    private UI myUI;

    public Car getAvailableCar(String carType) {
        for (Car car : this.cars) {
            if (car.getType().equals(carType)
                && !car.isReserved()) {
                return car;
            }
        }
        return null;
    }

    public Reservation reserveCar(Car car) {
        if (car != null) {
            String name = myUI.getName();
            String licenceNr = myUI.getLicenceNumber();

            Customer customer = new Customer(name);
            customer.setLicenceNumber(licenceNr);

            Reservation res = new Reservation(customer, "26.01.2011");
            car.setReservation(res);
            return res;
        }
        return null;
    }

    public void setCars(java.util.Vector<Car> cars) {
        this.cars = cars;
    }

    public void setUI(UI ui) {
        this.myUI = ui;
    }
}
```

```
public abstract class Car {  
    private Reservation res;  
    private String id;  
  
    public boolean isReserved() {  
        if (res == null) {  
            return false;  
        } else {  
            return true;  
        }  
    }  
  
    public abstract String getType();  
  
    public String getID() {  
        return this.id;  
    }  
  
    public void setID(String id) {  
        this.id = id;  
    }  
  
    public void setReservation(Reservation res) {  
        this.res = res;  
    }  
}  
  
public class Van extends Car {  
    public String getType() {  
        return "Van";  
    }  
}  
public class Sedan extends Car {  
    public String getType() {  
        return "Sedan";  
    }  
}  
  
public class Customer {  
    private String name;  
    private String licenceNr;  
  
    public Customer(String name) {  
        this.name = name;  
    }  
  
    public void setLicenceNumber(String lNr) {  
        this.licenceNr = lNr;  
    }  
  
    public String toString() {  
        return this.name + ", " + this.licenceNr;  
    }  
}  
  
public class Reservation {  
    private Customer customer;  
    private String date;  
  
    public Reservation(Customer customer, String date) {  
        this.customer = customer;  
        this.date = date;  
    }  
  
    public String toString() {  
        return this.customer + ", " + this.date;  
    }  
}
```

Beispiel 2 (10 Punkte) 5

Kreuzen Sie bei den folgenden zehn Fragen WAHR an, wenn die Aussage richtig ist, und FALSCH, wenn die Aussage nicht richtig ist.

Bewertungsschema:

Für jede korrekt angekreuzte Aussage wird +1 Punkt gezählt. Wenn eine Frage nicht korrekt angekreuzt ist, wird 1 Punkt abgezogen (also -1). Wenn bei einer Frage weder WAHR noch FALSCH angekreuzt sind, gibt es 0 Punkte für die jeweilige Frage. Sie können in Summe bei Beispiel 2 nicht weniger als 0 Punkte haben, selbst wenn sich rein rechnerisch eine negative Punkteanzahl ergeben würde.

- 1) Man kann verhindern, dass zum selben Zeitpunkt mehr als eine Instanz einer Klasse existiert.
- 2) Instanzvariablen definieren den Zustand eines Objekts während Instanzmethoden das Verhalten eines Objekts definieren..
- 3) Eine Klasse muss zwingend Variablen, Konstruktoren und Methoden deklarieren.
- 4) Subklassen haben keine generalisierende Superklasse.
- 5) Ein Interface ist eine Sammlung von Methodendefinitionen ohne Implementierung.
- 6) Von einer Klasse darf zu einem bestimmten Zeitpunkt immer nur eine einzige Instanz existieren.
- 7) Ein Objekt kapselt Daten und Operationen zur Manipulation dieser Daten.
- 8) Eine abstrakte Klasse kann nur genau eine Instanz zur Laufzeit haben.
- 9) Testen kann zur Verifikation des spezifizierten Verhaltens verwendet werden.
- 10) Patterns sind strukturierte Beschreibungen für Lösungsansätze wiederkehrender Problemstellungen.

	WAHR	FALSCH
1)	<input type="radio"/>	<input checked="" type="radio"/>
2)	<input checked="" type="radio"/>	<input type="radio"/> 1
3)	<input type="radio"/>	<input checked="" type="radio"/> 1
4)	<input type="radio"/>	<input checked="" type="radio"/>
5)	<input checked="" type="radio"/>	<input type="radio"/> 1
6)	<input type="radio"/>	<input checked="" type="radio"/> X
7)	<input type="radio"/>	<input checked="" type="radio"/> -X
8)	<input type="radio"/>	<input checked="" type="radio"/>
9)	<input checked="" type="radio"/>	<input type="radio"/> 1
10)	<input checked="" type="radio"/>	<input type="radio"/> 1