

برنامه‌سازی پیشرفته



برنامه‌نویسی شی‌گرا

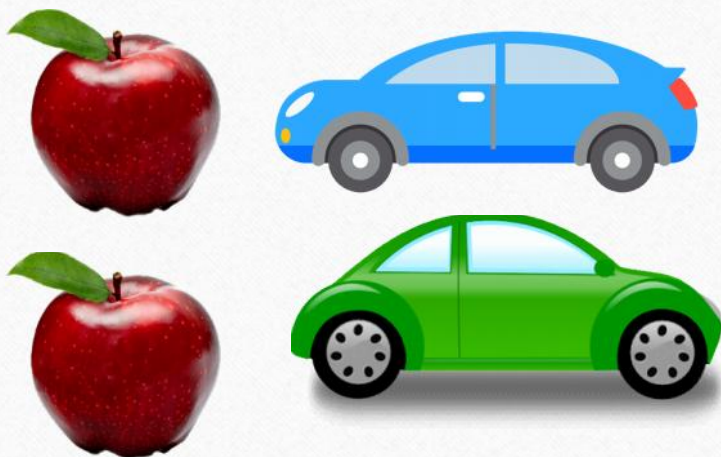
مهدی مصطفی زاده

سرفصل مطالب

- شیء و رده
- سطوح دسترسی
- کلیدواژه‌ی this
- ارکان برنامه‌نویسی شیء‌گرا



جهان واقع، مجموعه‌ای از اشیاء



- در زیر چند شیء می‌بینید؟

- هر یک را توصیف نمایید.

- هر یک چه رفتارهایی می‌تواند از خود نشان دهد؟

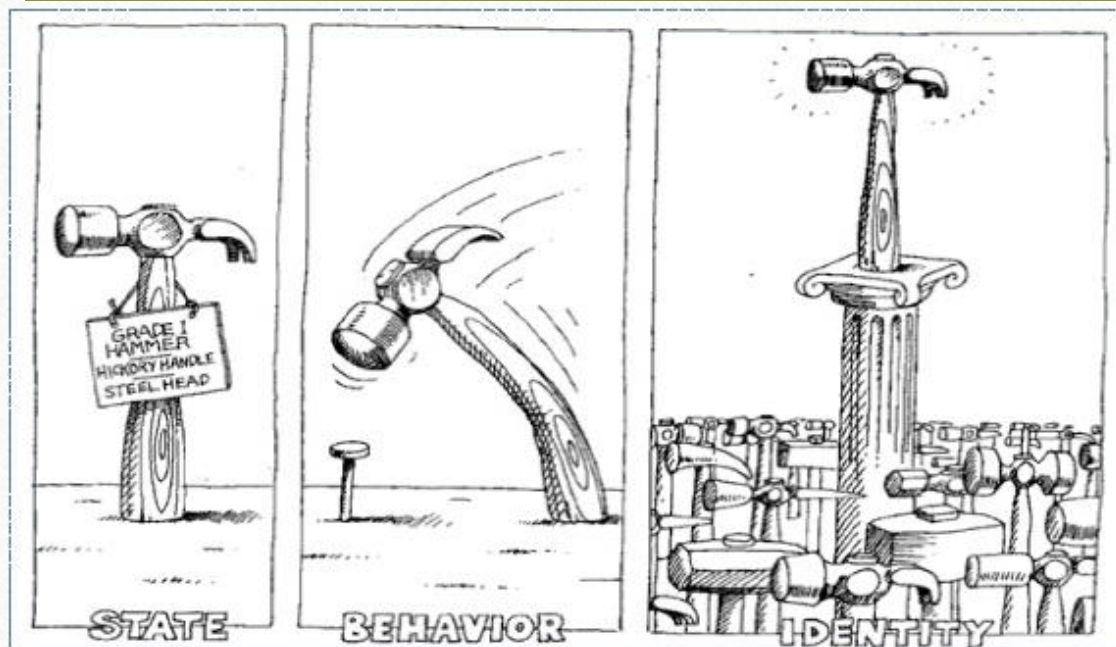
چه دستوراتی می‌توان به هر یک داد؟

چه تغییری می‌توان در حالت و وضعیت هر یک به وجود آورد؟

- این اشیاء را چگونه دستهبندی می‌کنید؟



شیء چیست؟



• هر شیء دارای

• حالت

• رفتار

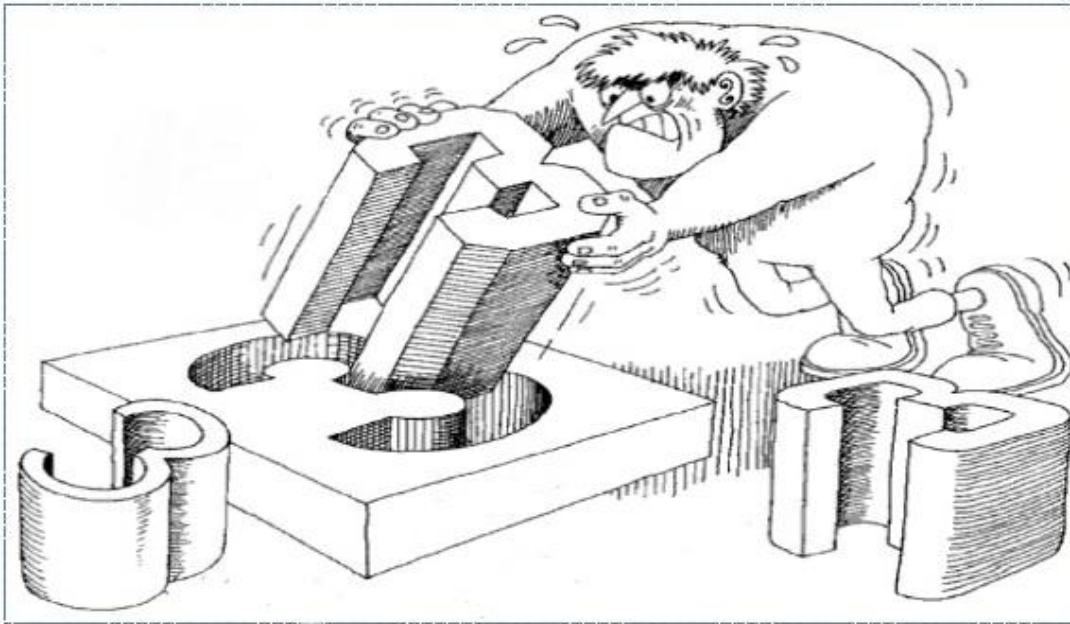
• هویت

است



رده چیست؟

- هر رده، مجموعه‌ای از اشیاء را نشان می‌دهد که ساختار و رفتار یکسان دارند.



مثال (برنامه‌ی مدیریت سفارش‌ها)

- می‌خواهیم برنامه‌ی مدیریت سفارش‌های یک رستوران فرضی را پیاده‌سازی کنیم.
- هر سفارش دارای حداکثر 10 قلم جنس خواهد بود.
- هر سفارش بایستی قابلیت افزودن قلم جنس جدید را داشته باشد.
- اگر ظرفیت سفارش پر باشد، پیغام «Order is full» نمایش داده شود.




```

1 package main;
2
3 import order.management.*;
4
5 public class Main {
6
7     public static void main(String[] args) {
8         Order order = new Order();
9     }
10
11 }

```

The type Order is not visible

1 quick fix available:

[Change visibility of 'Order' to 'public'](#)

Press 'F2' for focus

```

public static void main(String[] args) {
    Order order = new Order();
    order.addItem("Apple");
}

```

The method addItem(String) from the type Order is not visible

1 quick fix available:

[Change visibility of 'addItem\(\)' to 'public'](#)

Press 'F2' for focus

```

1 package order.management;
2
3 class Order {
4
5     String[] items = new String[10];
6     int numberOfItems = 0;
7
8     void addItem(String item) {
9         if(numberOfItems == 10) {
10             System.out.println("Order is full!");
11             return;
12         }
13         items[numberOfItems++] = item;
14     }
15
16 }

```



سطوح دسترسی (Access levels)

- کلیدواژه‌های تعیین‌کننده‌ی دسترسی (Access Modifier یا Access Specifier):

در سطح اعضای رده	در سطح رده	
دسترسی از همه جا		public
دسترسی از درون همان رده	فعلاً موضوعیت ندارد.	private
دسترسی از درون همان بسته		package-access
بعداً در مورد آن صحبت خواهیم کرد.		protected

- هر فایل در جاوا، حداکثر یک رده‌ی عمومی (public) دارد که حتماً پایستی هم‌نام فایل باشد.



مثال (برنامه‌ی مدیریت سفارش‌ها) – ادامه

```
Order.java Main.java
1 package order.management;
2
3 public class Order {
4
5     String[] items = new String[10];
6     int numberOfItems = 0;
7
8     public void addItem(String item) {
9         if(numberOfItems == 10) {
10             System.out.println("Order is full!");
11             return;
12         }
13         items[numberOfItems++] = item;
14     }
15
16 }
```

- برنامه‌ای که نوشته‌ایم، بسیار ناقص است!

- به ازای هر قلم جنس، مقدار، مبلغ واحد و
مبلغ کل آن مشخص نیست!



مثال (برنامه‌ی مدیریت سفارش‌ها) – ادامه

```
Order.java Main.java Item.java
1 package order.management;
2
3 public class Item {
4     private String title;
5     private int quantity;
6     private int unitPrice;
7
8     public String getTitle() {
9         return title;
10    }
11
12    public void setTitle(String title) {
13        this.title = title;
14    }
15
16    // other getters & setters
17 }
```

- مفهوم لقایف‌بندی (Encapsulation)

- مفهوم setter و getter

```
3 public class Order {
4     Order order = new Order();
5     Item item1 = new Item();
6     Item[] items = new Item[10];
7     int numberOfItems = 0;
8     public void addItem(Item item) {
9         if(numberOfItems == 10) {
10            System.out.println("Order is full!");
11            return;
12        }
13        items[numberOfItems++] = item;
14    }
15 }
```



سازنده (Constructor)

```
Order order = new Order();  
Item item1 = new Item();  
item1.setTitle("Apple");  
item1.setQuantity(2);  
item1.setUnitPrice(5_000);  
order.addItem(item1);
```

- در خط دوم کد مقابل، یک شیء از ردهی Item ایجاد شده است؛ اما تا پیش از اجرای سه خط بعد، حالت معتبری ندارد.

```
public Item(String t, int q, int u) {  
    title = t;  
    quantity = q;  
    unitPrice = u;  
}
```

- راه حل: استفاده از سازنده مناسب برای ردهی Item

```
public static void main(String[] args) {  
    Order order = new Order();  
    Item item1 = new Item("Apple", 2, 5_000);  
    order.addItem(item1);  
}
```

```
new Item();
```

The constructor Item() is undefined



مقداردهی اولیه به اشیاء

- چاوا، سه روش برای مقداردهی اولیه به حالت اشیاء، در نظر گرفته است:

```
Item[] items = new Item[10];  
int numberOfItems = 0;
```

```
Item[] items;  
int numberOfItems;  
  
{  
    items = new Item[10];  
    numberOfItems = 0;  
}
```

- مقداردهی اولیه در خط (Inline Initialization)

- بلوک مقداردهی اولیه (Initialization Block)

- سازنده (Constructor)

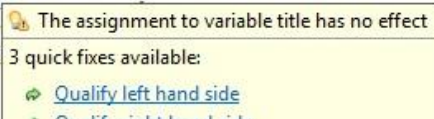


متغیر محلی هم نام با فیلد

```
public Item(String t, int q, int u) {  
    title = t;  
    quantity = q;  
    unitPrice = u;  
}
```

- سازنده‌ی فوق را به شکل زیر بازنویسی می‌کنیم:

```
public Item(String title, int quantity, int unitPrice) {  
    title = title;  
    quantity = quantity;  
    unitPrice = unitPrice;  
}
```



- راه‌حل: استفاده از کلیدواژه‌ی this:

```
public Item(String title, int quantity, int unitPrice) {  
    this.title = title;  
    this.quantity = quantity;  
    this.unitPrice = unitPrice;  
}
```



تعریف چند سازنده

```
public Item(String title, int quantity, int unitPrice) {  
    this.title = title;  
    this.quantity = quantity;  
    this.unitPrice = unitPrice;  
}
```

- علاوه بر سازنده‌ی فوق، سازنده‌ی زیر را نیز تعریف می‌کنیم:

```
public Item(String title, int unitPrice) {  
    this.title = title;  
    this.quantity = 1;  
    this.unitPrice = unitPrice;  
}
```

- بازپره‌پدی با استفاده از کلیدواژه‌ی this:

```
public Item(String title, int unitPrice) {  
    this(title, 1, unitPrice);  
}
```



کلیدواژه‌ی this

```
Order order = new Order();
order.addItem(new Item("Apple", 2, 5_000));
order.addItem(new Item("Orange", 3, 8_000));
order.addItem(new Item("Banana", 2, 10_000));
```

```
Order order = new Order();
order.addItem(new Item("Apple", 2, 5_000))
.addItem(new Item("Orange", 3, 8_000))
.addItem(new Item("Banana", 2, 10_000));
```

```
public Order addItem(Item item) {
    if(numberOfItems == 10) {
        System.out.println("Order is full!");
        return this;
    }
    items[numberOfItems++] = item;
    return this;
}
```

- سه کاربرد برای کلیدواژه‌ی this:

- ارجاع به شیء جاری
- بازپه‌پری از سازنده‌ها
- فراخوانی زنجیره‌ای متدها (کاربرد خاص از مورد اول)



مقداردهی اولیه به فیلدهای ایستا

- چاوا، دو روش برای مقداردهی اولیه به حالت اشیاء، در نظر گرفته است:

```
static float PI = 3.14f;  
  
static {  
    PI = 3.14f;  
}
```

- مقداردهی اولیه در خط (Inline Initialization)
- بلوک ایستا (Static Block)



تکمیل مثال

```
public class Order {
    String customer;
    Item[] items;
    int numberOfItems = 0;

    public Order(String customer, int capacity) {
        this.customer = customer;
        items = new Item[capacity];
    }

    public Order addItem(Item item) {
        if(numberOfItems == items.length) {
            System.out.println("Order is full!");
            return this;
        }
        items[numberOfItems++] = item;
        return this;
    }

    public int getTotalPrice() {
        int result = 0;
        for(int i = 0; i < numberOfItems; i++)
            result += items[i].getPrice();
        return result;
    }
}
```

```
public class Item {
    private String title;
    private int quantity;
    private int unitPrice;

    public Item(String title, int quantity, int unitPrice) {
        this.title = title;
        this.quantity = quantity;
        this.unitPrice = unitPrice;
    }

    public Item(String title, int unitPrice) {
        this(title, 1, unitPrice);
    }

    public int getPrice() {
        return quantity * unitPrice;
    }

    // getters & setters
}
```



الگوی طراحی Singleton

- مسأله: می‌خواهیم یک رده (Class) را به گونه‌ای پیاده‌سازی کنیم که در کل سامانه، فقط

یک شیء از آن داشته باشیم.



مثالی از الگوی طراحی Singleton

```
1 package lib;
2
3 public class Library {
4
5     private static Library library;
6
7     private Library() {
8
9     }
10
11     public static Library getInstance() {
12         if(library == null)
13             library = new Library();
14         return library;
15     }
16
17 }
```

```
public static void main(String[] args) {
    Library library = new Library();
}
```

The constructor Library() is not visible
1 quick fix available:

```
public static void main(String[] args) {
    Library library = Library.getInstance();
    Library anotherlibrary = Library.getInstance();
    System.out.println(library == anotherlibrary); // true
}
```



رابطه با خود (Self-Relationship)

```
package family;

public class Person {

    private String name;
    private int age;
    private Person father;

    public Person(String name, int age, Person father) {
        this.name = name;
        this.age = age;
        this.father = father;
    }

    public Person(String name, int age) {
        this(name, age, null);
    }

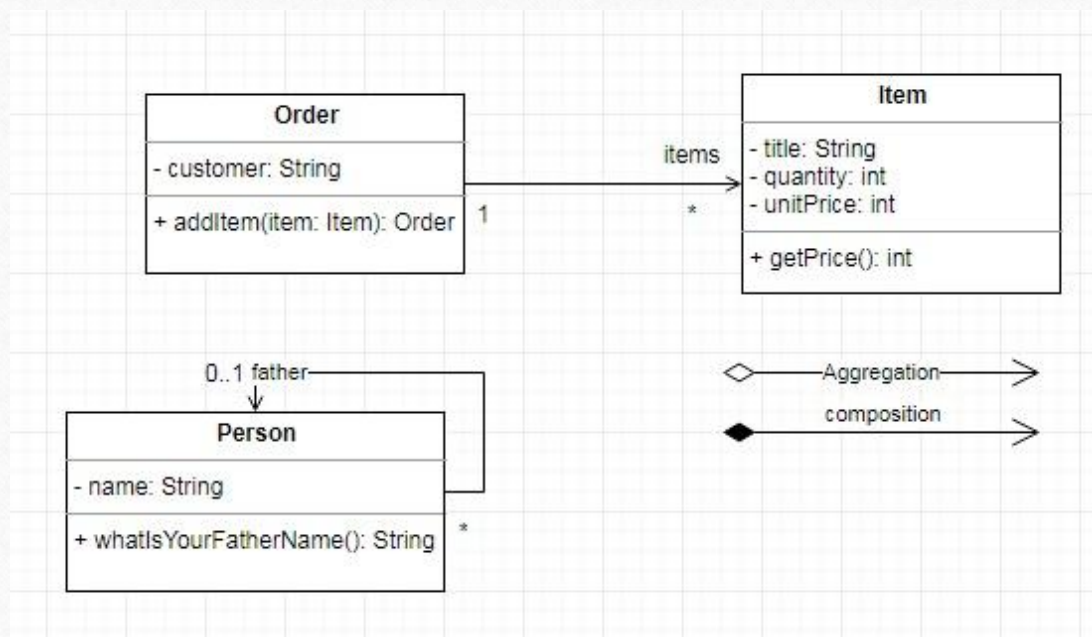
    public String whatIsYourFatherName() {
        return father == null ? "-" : father.name;
    }
}
```

- یک شیء می تواند با شیء یا اشیائی از جنس خودش رابطه داشته باشد.

- رابطه ی خویشاوندی افراد
- رابطه ی پیشنهادی و همسازی بین دروس
- ...



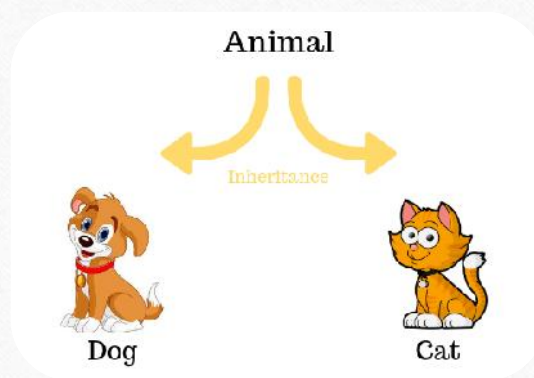
نمودار کلاس UML



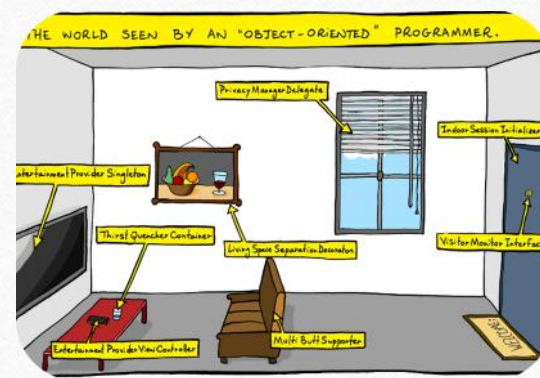
ارکان برنامه نویسی شیء گرا



چندریختی



وراثت



لفافه بندی

