

به نام پروردگار دانایی

# طراحی سیستم‌های شی گرا

برنامه‌نویسی شی گرا

---

درس اول: تعریف کلاس

سید کاوه احمدی

# مرور مفاهیم پایه

- شی (Object)
- کلاس (Class)
- فیلد (fields)
- سازنده‌ها (constructors)
- متد (method)
- پارامتر (parameter)
- نوع داده‌ای (data type)

■ شی یک مفهوم کلی است به طوریکه دارای هویت بوده و می تواند وضعیت خود را ثبت کند و همچنین رفتارهایی از خود بروز دهد.

— **هویت (Identity):** آن ویژگی از یک شی است باعث می شود یک شی از تمام اشیا دیگر متمایز شود.

— **وضعیت (State):** هر شی دارای صفاتی است. مقادیر تمام صفات یک شی (ایستا) وضعیت شی را مشخص می کند.

■ وضعیت شی، نتیجه تجمعی رفتارهای شی را نمایش می دهد.

— **رفتار (Behavior):**

■ رفتارها وضعیت شی را تغییر می دهند.

■ چگونگی عمل و عکس العمل شی در مقابل دریافت یا ارسال پیام. (تعامل با سایر اشیا).

- مشخص کننده نوع شی.

- **مشخص کننده صفات و رفتارهایی که یک شی دارد.**

- یک شی یک نمونه از یک کلاس است.

– از روی یک کلاس می توان اشیای مختلفی ایجاد کرد که همه ی آنها دارای صفات و رفتارهای مشخص شده در کلاس هستند اما وضعیت آنها (مقادیری که صفات دارند) با یکدیگر متفاوت است.

- یک کلاس در واقع یک تجرید است، یک روش برای دسته بندی اشیای مشابه.

کلاس = هویت + صفات + رفتار

شی = هویت + صفات + مقادیر صفات + رفتار

شی = کلاس + مقادیر صفات

- یک شی تجرید الگوریتم (نسل دوم زبان‌های برنامه‌سازی) را با تجرید داده‌ای (نسل سوم زبان‌ها) ترکیب می‌کند

# DEMO

# شی و کلاس

## ■ شی

- نمایش «چیزها»ی دنیای واقعی یا از دامنه‌ی یک مسئله
- ماشین قرمز که آن پایین در پارکینگ پارک کرده.

## ■ کلاس

- نمایش تمام اشیای هم نوع
- ماشین

# متد و پارامتر

- اشیا دارای عملیاتی هستند احضار (`invoke`) شوند.

- به آنها متد گفته می شود.

- رفتارهای شی را پیاده سازی می کند.

- متدها ممکن است دارای پارامترهایی برای دریافت اطلاعات اضافی که برای اجرا به

- آنها نیاز دارد باشد.



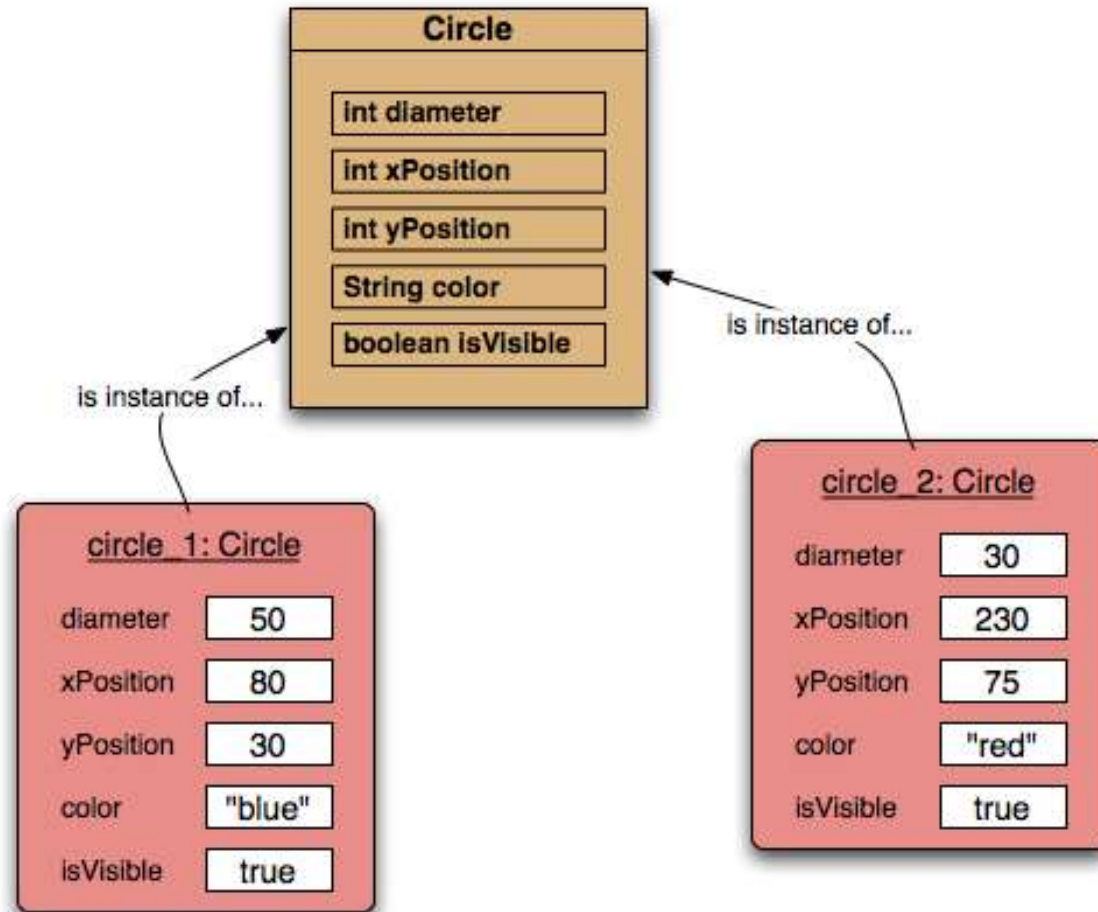
# مشاهدات دیگر

- نمونه‌های بسیاری از روی یک کلاس می‌توان ایجاد کرد.
- اشیا دارای خصوصیتی هستند: مقادیری که در فیلدها ذخیره می‌شوند.
- کلاس مشخص می‌کند شی دارای چه فیلدهایی است اما هر شی مقادیر ویژه خود را دارد (وضعیت شی).

circle1 : Circle

private int diameter	68	Inspect
private int xPosition	230	Get
private int yPosition	130	
private String color	"blue"	
private boolean isVisible	true	

Show static fields Close



# متن برنامه (Source Code)

- هر کلاس دارای یک متن برنامه است (کد جاوا) که جزئیات آنرا تعریف می کند (فیلدها و متدها).

## مقادیر بازگشتی (Return values)

- متدها می‌توانند نتیجه را به صورت مقدار بازگشتی، بازگردانند.
- این متدها دارای مقدار بازگشتی غیر `void` است.
- متدهای بدون مقدار بازگشتی، دارای نوع بازگشتی `void` هستند

# دستگاه فروش بلیط – نگاه بیرونی

■ بررسی رفتار یک دستگاه فروش بلیط معمولی

– پروژه *naive-ticket-machine* را ببینید.

– دستگاه بلیط‌های با قیمت ثابت عرضه می‌کند.

■ قیمت بلیط‌ها چگونه مشخص می‌شود؟

– پول چگونه وارد دستگاه می‌شود؟

– دستگاه چگونه پول‌های وارد شده را پیگیری می‌کند؟

# DEMO

# دستگاه فروش بلیط – نگاه درونی

- تعامل با یک دستگاه سرنخ‌هایی از رفتارهای آن به ما می‌دهد.
- نگاه درونی به ما کمک می‌کند بفهمیم این رفتار چگونه ایجاد یا پیاده‌سازی شده است.
- تمام کلاس‌های جاوا دارای نمایش درونی مشابهی هستند.



# ساختار پایه کلاس

```
public class TicketMachine  
{  
    Inner part omitted.  
}
```

شکل بیرونی کلاس TicketMachine

```
public class ClassName  
{  
    Fields  
    Constructors  
    Methods  
}
```

محتویات درون یک کلاس

# کلمات کلیدی (Keywords)

■ کلمات با معنای خاص در زبان برنامه‌نویسی

- `public`
- `class`
- `private`
- `int`

■ به آنها کلمات رزرو شده نیز گفته می‌شود.

```
public class TicketMachine
{
    private int price;
    private int balance;
    private int total;

    Further details omitted.
}
```

visibility modifier      type      variable name

↙                      ↓                      ↘

private int price;

- فیلدها مقادیر شی‌ها را ذخیره می‌کند.
- به عنوان متغیرهای یک نمونه نیز شناخته می‌شوند.
- برای مشاهده وضعیت شی مورد بررسی قرار می‌گیرند.
- فیلدها وضعیت شی را مشخص می‌کند.
- مقادیر برخی فیلدها به ندرت تغییر می‌کنند و برخی غالباً نیاز به تغییر دارند.
- نام کلاس‌ها را با حروف بزرگ شروع کنید نام فیلدها را با حروف کوچک.

## معرف دسترسی (visibility modifier) (بیان غیر دقیق)

- `public`: فیلد یا متد می تواند از همه جا فراخوانی شود.
- `private`: فراخوانی فقط از همان کلاس.
  - به ما کمک می کند پیاده سازی ها را از بیرون پنهان کند.
- فیلدها معمولاً `private` هستند. سازنده ها معمولاً `public` و متدها فقط وقتی که ضروری است باید `public` تعریف شوند.
- مشخص کننده سطح دسترسی
- جاوا همچنین سطوح دسترسی `protected` و `package` نیز دارد. (در آینده)

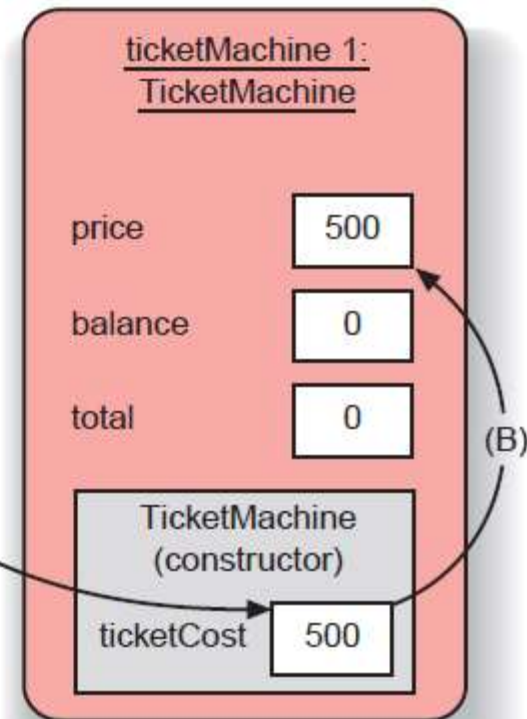
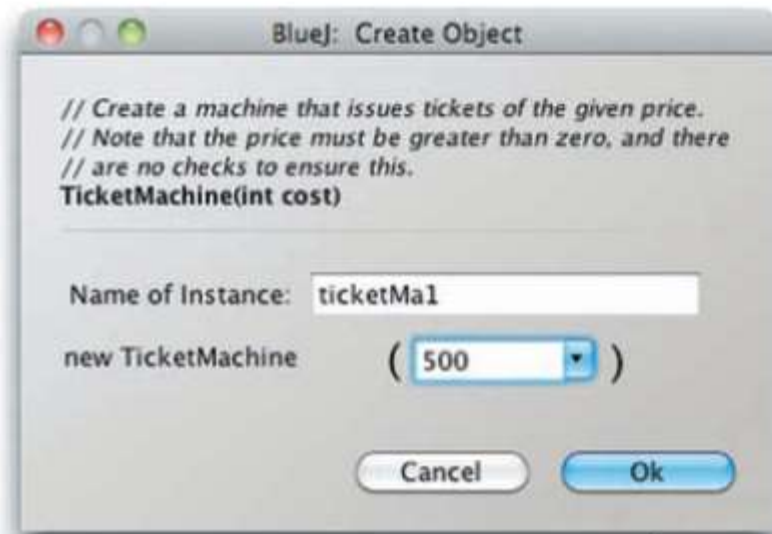
# سازنده‌ها (Constructors)

- یک شی را مقداردهی اولیه می‌کند.
- همنام با کلاس هستند.
- ارتباط نزدیکی با فیلدها دارد.
- برای ذخیره‌سازی مقادیر اولیه فیلدها مورد استفاده قرار می‌گیرند.
- برای این کار از پارامترها استفاده می‌کنند.

```
public class TicketMachine
{
    ...
    public TicketMachine(int cost)
    {
        price = cost;
        balance = 0;
        total = 0;
    }
    ...
}
```

**Constructor**

# ارسال اطلاعات از طریق پارامترها



پارامترها نوع دیگری از متغیرها هستند.  
(همانند فیلدها)

## انتساب (Assignment)

- مقادیر در فیلدها (و همچنین سایر متغیرها) به وسیله عملگر انتساب ذخیره می‌شوند.
  - *variable = expression;*
  - `price = 5 ;`
  - `price = cost;`
- یک متغیر فقط می‌تواند یک مقدار را ذخیره کند. بنابراین مقدار قبلی از بین خواهد رفت.

# انتخاب نام متغیرها

- آزادی فراوانی در انتخاب نام متغیرها وجود دارد. از آن عاقلانه استفاده کنید!
- از نام‌های معنادار استفاده کنید تا متن برنامه قابل فهم‌تر شود.
  - price, amount, name, age, **etc.**
- از نام‌های تک کاراکتری یا مرموز اجتناب کنید.
  - w, t5, xyz123



# متدها (Methods)

- متدها رفتارهای یک شی را پیاده‌سازی می‌کنند.

- متدها دارای ساختاری مشابه هستند: یک سرآیند (header) و یک بدنه (body)

- متدهای دسترسی (Accessor methods) اطلاعاتی در مورد شی را ارائه می‌کنند.

- متدهای تغییر (Mutator methods) وضعیت یک شی را تغییر می‌دهند.

- انواع دیگر متدها برای انجام کارهای دیگر مورد استفاده قرار می‌گیرد.

# متدها (Methods)

- سرآیند (header)، شناسنامه متد است!

```
public int getPrice()
```

- سرآیند به ما می گوید:

- نام متد

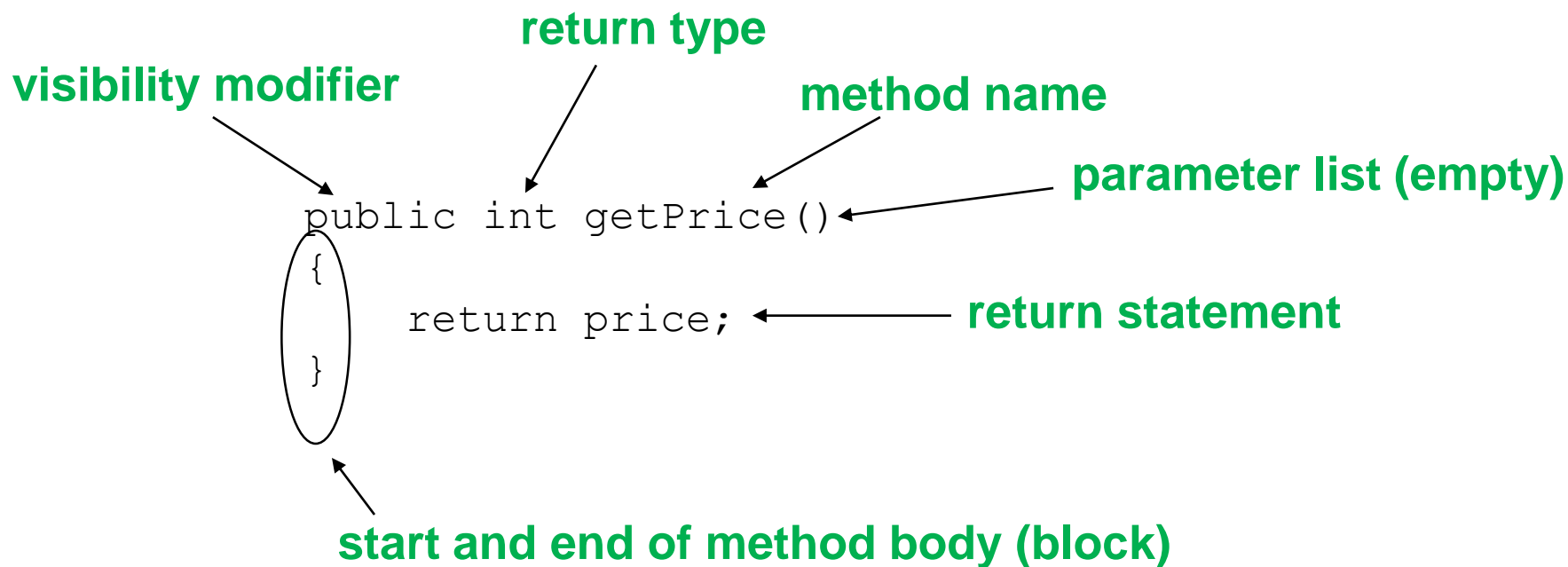
- چه پارامترهایی می گیرد

- چه نتیجه‌ای باز می گرداند

- دسترسی سایر اشیا و کلاس‌ها به آن

- بدنه پیاده‌سازی متد را در بر می گیرد.

# متدهای دسترسی (Accessor (**get**) methods)



# متدهای دسترسی

- یک متد دسترسی همیشه نوع بازگشتی‌ای غیر از `void` دارند.
- یک متد دسترسی یک مقدار (نتیجه) که در سرآیند مشخص شده را باز می‌گرداند.
- متد دارای یک عبارت `return` برای بازگرداندن مقدار است.
- بازگرداندن به معنی چاپ کردن نیست!
- به آنها متدهای `getter` نیز گفته می‌شود.

■ اشکالات را بشمارید

— پنج مورد

```
public class CokeMachine
{
    int
    private price;

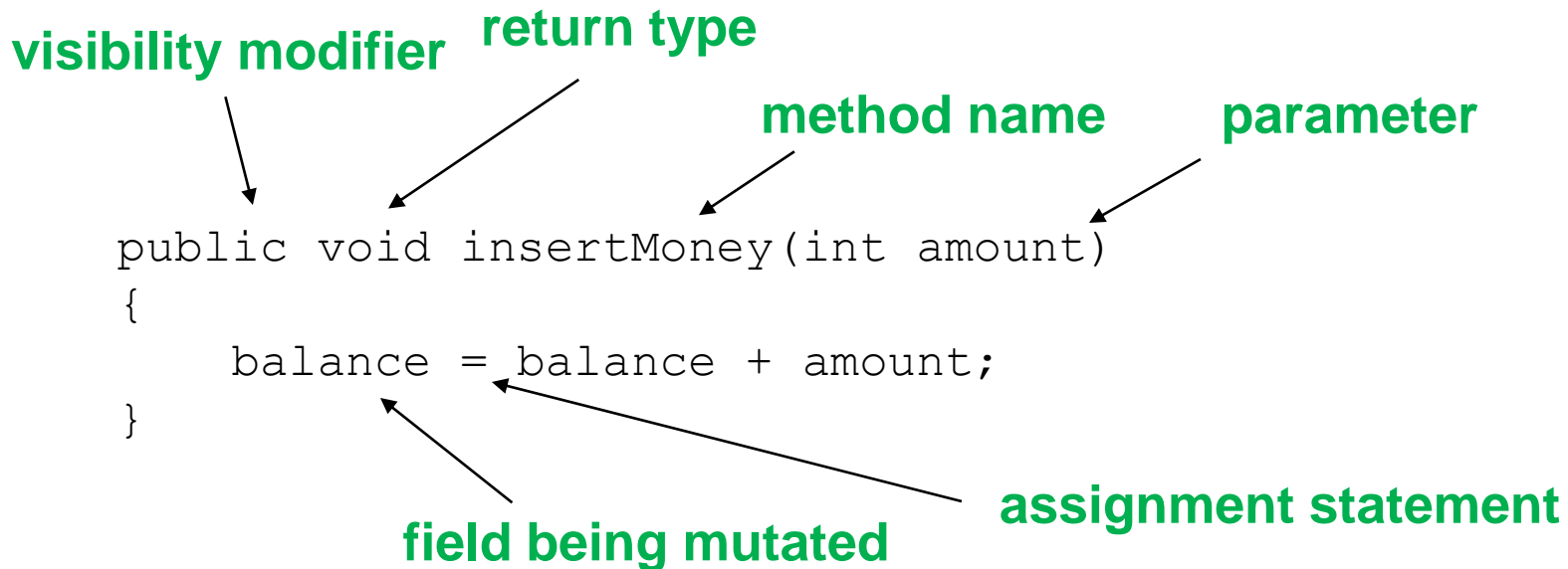
    public CokeMachine()
    {
        price = 300;
    }

    public int getPrice()
    {
        return Price;
    }
}
```

# متدهای تغییر (Mutator methods)

- ساختار مشابه همه متدها: سرآیند و بدنه.
- برای تغییر دادن وضعیت شی مورد استفاده قرار می‌گیرد.
  - به وسیله تغییر دادن مقدار یک یا چند فیلد.
  - به طور معمول شامل عملگر انتساب.
  - معمولا پارامتر دریافت می‌کنند.

# متدهای تغییر (Mutator methods)



# متدهای تغییر set

- فیلدها معمولا متدهای تغییر (set) ویژه خود را دارند.

- شکل مشخص و ساده‌ای دارند:

- نوع بازگشتی void

- نام متد مرتبط به نام فیلد

- دارای یک پارامتر هم نوع با فیلد

- یک عملگر انتساب



## یک متد set مرسوم

```
public void setDiscount(int amount)
{
    discount = amount;
}
```

می‌توان نتیجه گرفت discount یک فیلد با نوع int است:

```
private int discount;
```

## تغییرات محافظت شده

- متد `set` نباید همواره پارامتر دریافتی را به فیلد منتسب کند.
- پارامتر ممکن است به لحاظ معتبر بودن مورد بررسی قرار گیرد و در صورت نامناسب بودن رد شود.
- متدهای تغییر، داشتن فیلدهای محافظت شده را میسر می کند.
- متدهای تغییر، از محصور سازی پشتیبانی می کند.

# ساختار کلاس تا کنون

```
public class TicketMachine
```

```
{
```

```
    private int price; ← تعریف فیلد (خصوصیت)
```

```
    public TicketMachine(int ticketCost)
```

```
    {
```

```
        price = ticketCost;
```

```
    }
```

مقداردهی اولیه به خصوصیت  
(سازنده با پارامتر ورودی)

```
    public int getPrice()
```

```
    {
```

```
        return price;
```

```
    }
```

```
}
```

دسترسی به مقدار خصوصیت

# چاپ در ترمینال

■ برای چاپ اطلاعات در پنجره ترمینال توسط جاوا، از متد `println` شی استاتیک `out` در کلاس `System` و به شکل زیر استفاده می‌کنیم:

– `System.out.println("Output Terminal String");`

■ کلاس `System` جزو کلاس‌های کتابخانه `java.lang` است که به طور پیش‌فرض به تمام کلاس‌ها الحاق می‌شود و نیازی به الحاق آن به صفحه نیست.

```
public void buyTicket()
{
    if(balance >= price) {
        // Simulate the printing of a ticket.
        System.out.println("#####");
        System.out.println("# Ticket");
        System.out.println("# " + price + " cents.");
        System.out.println("#####");
        System.out.println();

        // Update the total collected with the price.
        total = total + price;
        // Reduce the balance by the price.
        balance = balance - price;
    }
    else {
        System.out.println("You must insert at least: " +
            (price - balance) + " more cents.");
    }
}
```

# نکاتی از رشته و چاپ!

- تعریف یک متغیر از نوع رشته

- `String str = "Result";`

- رشته‌ها به وسیله دو علامت " مشخص می‌شوند.

- الحاق رشته‌ها توسط اپراتور + صورت می‌گیرد.

- الحاق هر نوع داده‌ای با یک رشته، منجر به تبدیل آن داده به رشته می‌شود.

- `4 + 5`

  - `9`

- `"wind" + "ow"`

  - `"window"`

➔ **overloading**

- `String str = "Result";`

  - `str + ": " + 6`

  - `"Result: 6"`

- `int price = 500;`

  - `"# " + price + " cents"`

  - `"# 500 cents"`

- `System.out.println(5 + 6 + "hello");`  
– `11hello`
- `System.out.println("hello" + 5 + 6);`  
– `hello56`



- متدها تمام رفتارهای شی را پیاده سازی می کنند.
- متد دارای یک نام و یک مقدار بازگشتی است.
  - مقدار بازگشتی ممکن است void باشد.
  - در صورتی که مقدار بازگشتی void نباشد، متد باید مقداری را به فراخواننده ی خود بازگرداند.
- متد ممکن است پارامترهای ورودی داشته باشد.
  - پارامترها مقادیری را از بیرون برای استفاده ی متد فراهم می کنند.

# تاملی بر دستگاه بلیط فروش

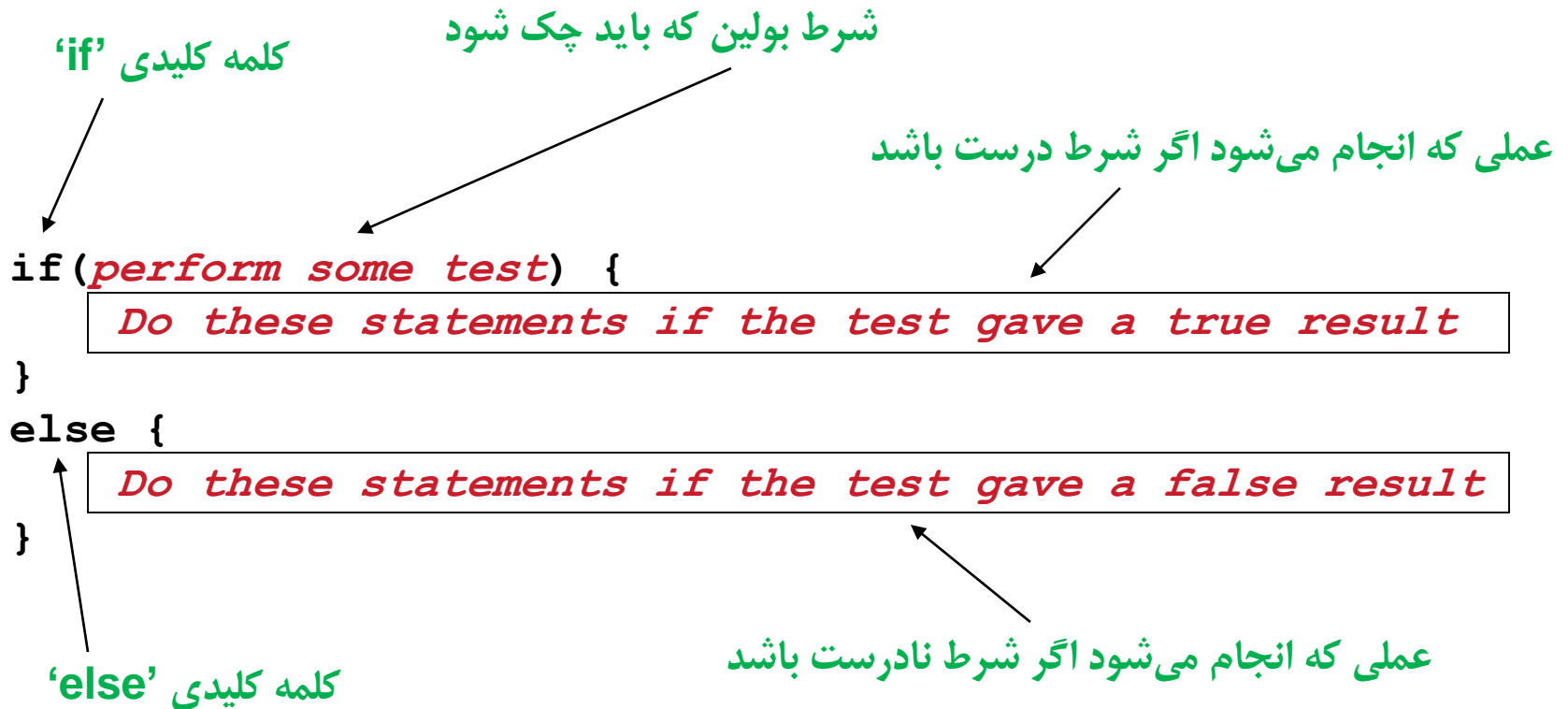
- رفتار آنها از چند جنبه نامناسب است:
  - هیچ کنترلی روی ورودی amount وجود ندارد.
    - می توان مقدار منفی وارد کرد.
    - بازپرداخت پول وجود ندارد.
    - هیچ کنترلی برای معقول بودن مقادیر اولیه وجود ندارد.
      - تعریف قیمت بلیط با یک مقدار منفی
  - چطور می توان بهتر این کار را انجام داد؟
    - نیاز به رفتارهای پیچیده تری داریم

# انتخاب در زندگی روزمره

- اگر مقدار کافی داشته باشم، برای خوردن غذا بیرون می‌روم.
- در غیر اینصورت خانه می‌مانم و فیلم می‌بینم.

```
if(I have enough money left) {  
    go out for a meal;  
}  
else {  
    stay home and watch a movie;  
}
```

# انتخاب در جاوا (دستور شرطی if)



# انتخاب در ماشین بلیط فروش

- معتبر سازی یک پارامتر (Validation)

```
public void insertMoney(int amount)
{
    if(amount > 0) {
        balance = balance + amount;
    }
    else {
        System.out.println("Use a positive amount: " +amount);
    }
}
```

متدی برای بازپرداخت پول (`refundBalance`):

چطور می‌توان متدی برای دریافت مابقی پول و  
صفر کردن `balance` نوشت؟

## نوشتن متد refundBalance

```
balance = 0;  
return balance;
```

- اشکال این کد این است که مقدار صفر را باز می‌گرداند نه balance مورد نظر را.

```
return balance;  
balance = 0;
```

- اشکال این کد این است که فراخوانی return منجر به خروج از متد می‌شود و balance صفر نمی‌شود.

- راه حل: استفاده از یک متغیر محلی!

## متغیرها – تا اینجا

- فیلدها یک نوع از انواع متغیرها است.
  - مقادیر را تا زمانی که شی وجود دارد نگهداری می کنند.
  - در سراسر کلاس قابل دسترسی هستند.
- پارامترها نوع دیگری از متغیرها هستند.
  - آنها مقادیرشان را از بیرون متد دریافت می کنند.
  - به متد کمک می کنند کار خود را کامل کند.
  - در هر بار فراخوانی، متد مجموعه جدیدی از پارامترها را دریافت می کند.
  - مقادیر پارامترها عمر کوتاهی دارند.



# متغیرهای محلی

■ متدها می‌توانند متغیرهای محلی ویژه خود را داشته باشند

— طول عمر کم همانند پارامترها.

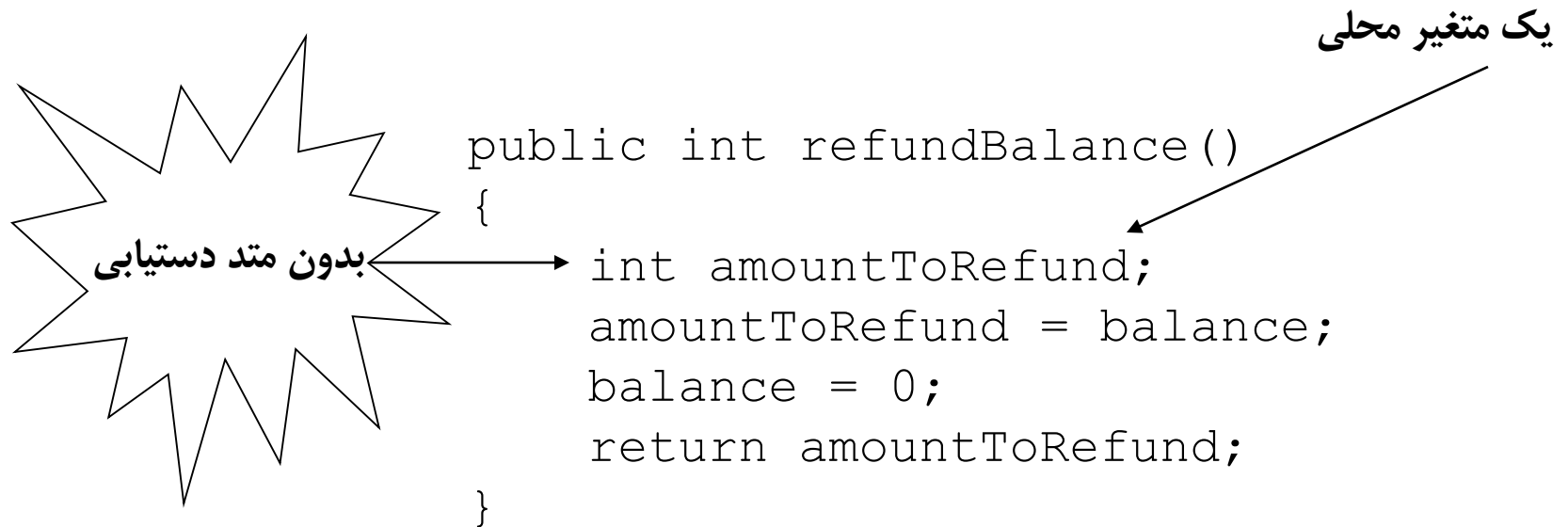
— متد مقدار آنها را تنظیم می‌کند. برخلاف پارامترها که مقدارشان از بیرون گرفته می‌شد.

— برای محاسبات و ذخیره‌سازی موقت مورد استفاده قرار می‌گیرد.

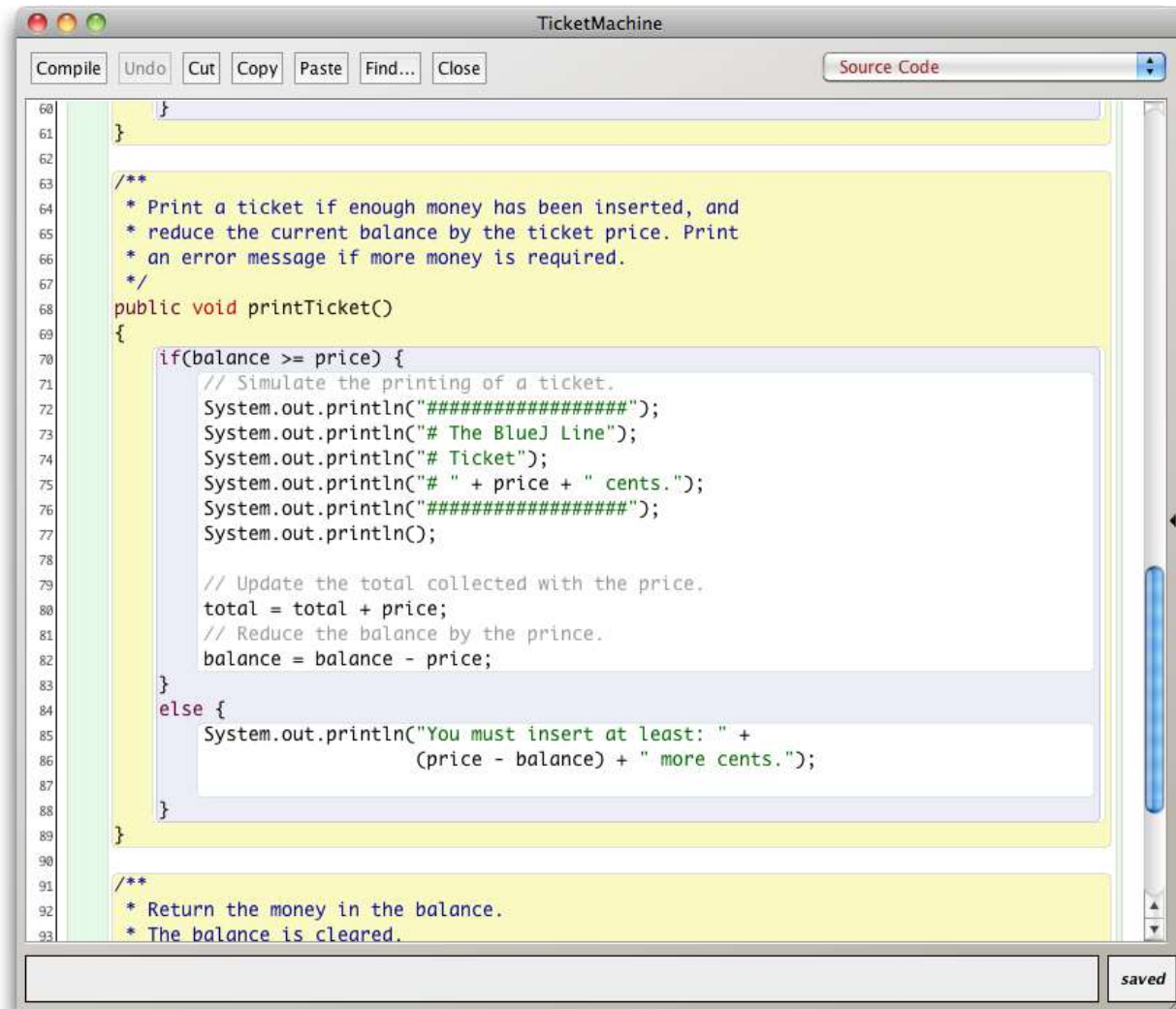
— فقط تا زمانی که متد در حال اجرا است وجود دارند.

— فقط داخل همان متد (یا بلوک تعریف شده) قابل دسترسی هستند.

# متد refundBalance



# مشخص کردن محدوده



```
60     }
61 }
62
63 /**
64  * Print a ticket if enough money has been inserted, and
65  * reduce the current balance by the ticket price. Print
66  * an error message if more money is required.
67  */
68 public void printTicket()
69 {
70     if(balance >= price) {
71         // Simulate the printing of a ticket.
72         System.out.println("#####");
73         System.out.println("# The BlueJ Line");
74         System.out.println("# Ticket");
75         System.out.println("# " + price + " cents.");
76         System.out.println("#####");
77         System.out.println();
78
79         // Update the total collected with the price.
80         total = total + price;
81         // Reduce the balance by the price.
82         balance = balance - price;
83     }
84     else {
85         System.out.println("You must insert at least: " +
86             (price - balance) + " more cents.");
87     }
88 }
89
90
91 /**
92  * Return the money in the balance.
93  * The balance is cleared.
```

# قلمرو (Scope) و طول عمر (Lifetime)

- هر بلوک یک قلمرو جدید ایجاد می کند.
  - کلاس، متد، دستورات
- قلمرو می تواند تو در تو باشد.
  - بلوک دستورات داخل بلاکی دیگر داخل بدنه متد داخل بدنه کلاس.
- قلمرو ایستا است (کد نوشته شده)
- طول عمر پویا است (زمان اجرا)

# قلمرو (Scope) و طول عمر (Lifetime)

- قلمرو یک متغیر محلی، بلوکی است که در آن تعریف شده است.
  - طول عمر یک متغیر محلی، زمان اجرای بلوکی است که در آن تعریف شده است.
- قلمرو یک فیلد حداقل کل کلاس آن است.
  - طول عمر یک فیلد، طول عمر شی‌ای است که به آن تعلق دارد.
- معرف‌های دسترسی می‌تواند قلمرو یک فیلد را عوض کند.
  - فیلدهای `public` قلمرو سراسری (`global`) دارند.

## بارگذاری (Overload) کردن سازنده‌ها و متدها

- می‌توان چند سازنده برای یک کلاس نوشت.
- همه سازنده‌ها هم‌نامند اما لیست پارامترهای آنها متفاوت است.
- به این بارگذاری گفته می‌شود.

```
public TicketMachine() {  
  
}  
  
public TicketMachine(int ticketCost) {  
  
}
```

# بارگذاری (Overload) کردن سازنده‌ها و متدها

- متدها را همانند سازنده‌ها می‌توان بارگذاری کرد.
  - متد بر اساس پارامترهای ورودی انتخاب می‌شود.
  - به ما امکان می‌دهد کارهای مشابه را با آرگومان‌های متفاوت انجام دهیم.
  - یک کار چند پیاده‌سازی / چند ریختی

## ■ کلاس

– فیلدها

– سازنده‌ها

– متدها

■ فیلدها مقادیر ذخیره شده‌ای هستند که وضعیت شی را مشخص می‌کنند.

■ سازنده‌ها شی را تعریف اولیه می‌کند.

■ متدها پیاده‌سازی رفتارهای شی هستند.

■ فیلدها، پارامترها و متغیرهای محلی همه متغیر هستند.



- شی
- کلاس
- متدها
- پارامتر
- فیلد
- سازنده‌ها
- نوع داده (عددی و رشته‌های)
- کامنت‌گذاری
- انتساب (=)
- بلوک
- قلمرو و طول عمر
- مقدار بازگشتی
- void
- انتساب ترکیبی (+=, -=)
- if
- چاپ