دانشکده مهندسی و علوم کامپیوتر

برنامه نویسی پیشرفته

وحیدی اصل

آشنایی با شیئ گرایی

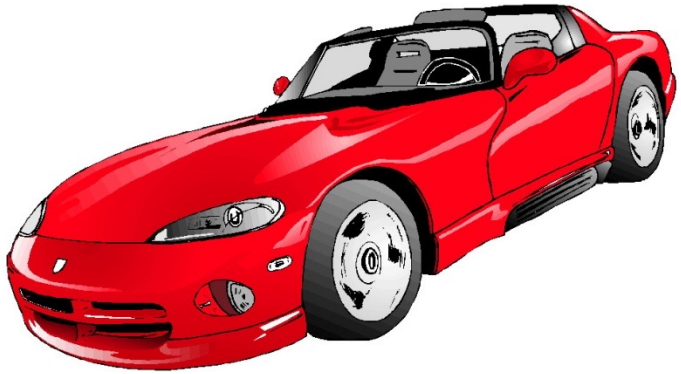
مباحث امروز

- آشنایی با مفهوم شئ گرایی
- تفاوت کلاس و شیئ
- نوشتن چندین کلاس ساده
- سازنده ها
- نحوه نمایش کلاسها و اشیا در حافظه
- نحوه ایجاد زباله
- متدها و متغیرهای نمونه
- متدها و فیلدهای استاتیک
- ارسال اشیا به متدها

مفاهیم برنامه نویسی شیئ گرا

•برنامه نویسی شیئ گرا (OOP) به معنای برنامه نویسی با استفاده از اشیا می باشد.

•یک شیئ بیانگر یک موجودیت در دنیای واقعی است که می تواند هویت مستقلی داشته باشد. برای مثال:



•یک دانشجو

•یک میز

•یک خودرو

•یک دکمه گرافیکی

•یک وام بانکی

• هر شیئ دارای هویت، حالت و رفتارهای خود است.

•حالت یک شیئ شامل مجموعه ای از فیلدهای داده ای با مقادیر آنها است.

• رفتار یک شیئ توسط مجموعه متدهای آن تعریف می شود.

یک شیئ دارای رفتار است!

- در شیوه های برنامه نویسی ساخت یافته (غیرشیئ گرا) داریم:
  - داده ها، که به صورت انفعالی (غیرفعال) در برنامه استفاده می شوند.
  - توابع، که قادرند برروی هر داده ای دستکاری انجام دهند.

- در شیوه برنامه نویسی شیئ گرا، برنامه از اشیا ساخته می شود. در درون هر شیئ (object) داده ها و متدهای مربوطه قرار می گیرند. این متدها برروی داده های همان شیئ دستکاری انجام می دهند.
  - یک شیئ فعال (active) است و قادر است کارهایی را انجام دهد.
  - یک شیئ مسئول داده های مربوط به خود است.
  - اما: می تواند داده های خود را برای دیگر اشیا در معرض نمایش و قابل استفاده قرار دهد.

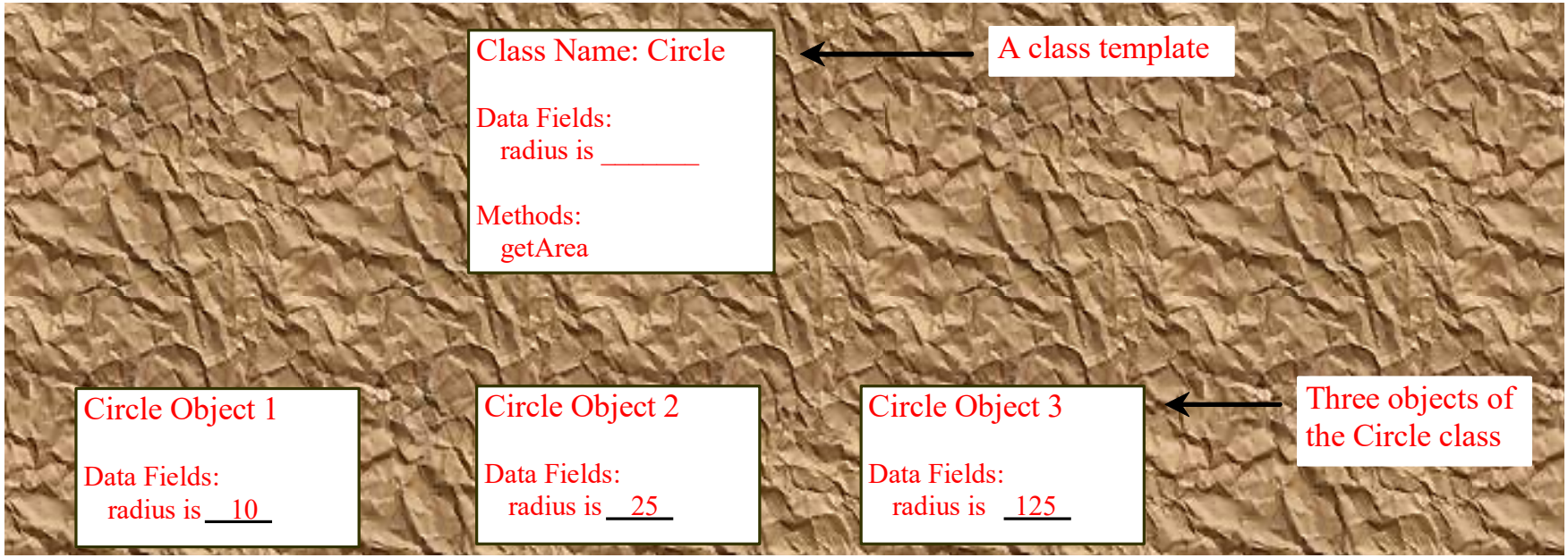
مثال: یک شیئ خرگوش

- شما می توانید (برای مثال در یک بازی) یک شیئ خرگوش ایجاد کنید.
  - این شیئ خرگوش می تواند داده هایی داشته باشد:
    - میزان گرسنگی آن را نشان دهد.
    - میزان ترسیدن آن را نشان دهد.
    - مکان فعلی آن را نشان دهد.
  - و متدهای زیر داشته باشد:
    - خوردن، پنهان شدن، کندن زمین، دویدن و ...

شیئی انسان

- فیلدها
  - عنوان
  - نام
  - نام خانوادگی
  - تاریخ تولد
  - آدرس
- عملیات
  - دانستن این صفات و تغییر دادن آنها
- ساده سازی برنامه نویسی←اضافه نمودن عملیات دانستن قد

اشیا



- یک شیئ هم دارای حالت و هم دارای رفتار است.
- حالت، تعریف کننده وضعیت شیئ بوده و رفتار می گوید آن شیئ می تواند چه کارهایی انجام دهد.

کلاسها

- کلاسها ساختارهایی هستند که اشیایی از یک نوع را تعریف می کنند.
- این نوع توسط کلاس مشخص می شود. کلاسها مانند قالبهایی هستند که اشیای مختلف از آنها ایجاد می شود.
- یک کلاس جاوا از متغیرها برای تعریف فیلدها و از متدها برای تعریف رفتارها استفاده می کند.
- علاوه بر این، یک کلاس شکل خاصی از متدها به نام سازنده ها (constructor) را فراهم می کند که به محض ایجاد یک شیئ از آن کلاس فراخوانی می شوند.

.

کلاسها

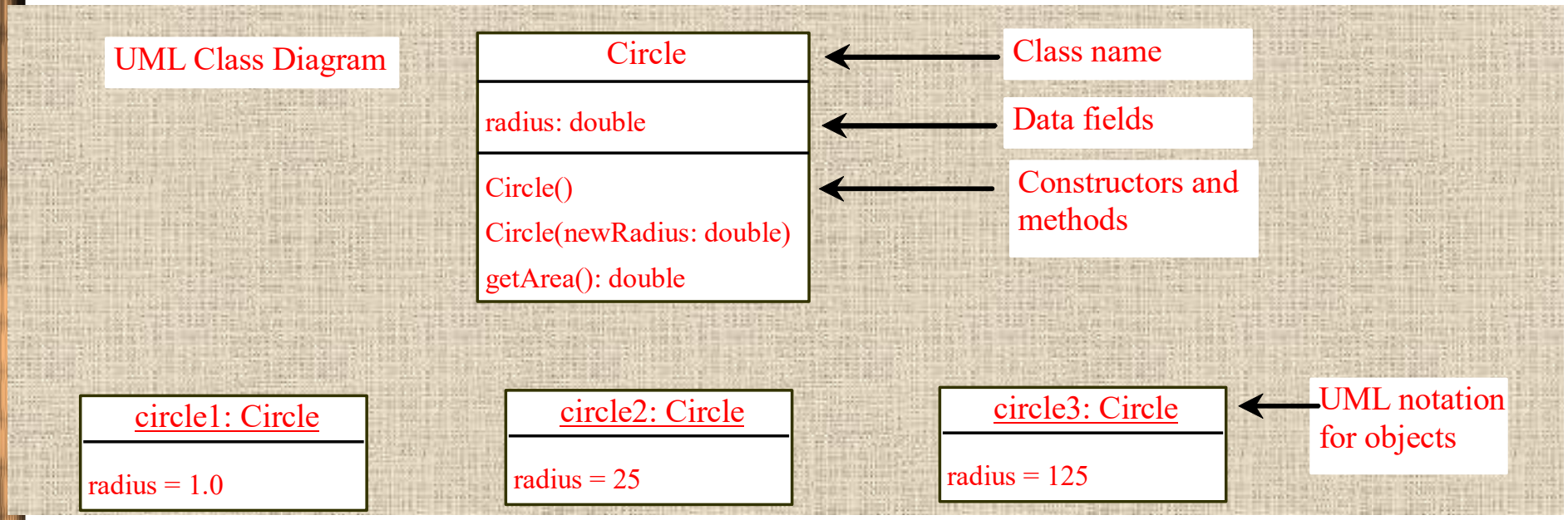
```
class Circle {
  /** The radius of this circle */
  double radius;

  /** Construct a circle object */
  Circle() {
  }

  /** Construct a circle object */
  Circle(double newRadius) {
    radius = newRadius;
  }

  /** Return the area of this circle */
  double getArea() {
    return radius * radius * 3.14159;
  }
}
```

Data field

Constructors

Method

کلاسها


UML Class Diagram
Circle
radius: double
Circle()
Circle(newRadius: double)
getArea(): double
Class name
Data fields
Constructors and methods
circle1: Circle
radius = 1.0
circle2: Circle
radius = 25
circle3: Circle
radius = 125
UML notation for objects

همیشه یک کلاس تعریف کنید و برای کلاس تعریفی، یک کلاس tester بسازید:

1 Write your class

```
class Dog {

  int size;
  String breed;
  String name;

  void bark() {
    System.out.println("Ruff! Ruff!");
  }
}
```

instance variables

a method

2 Write a tester (TestDrive) class

```
class DogTestDrive {
  public static void main (String[] args) {
    // Dog test code goes  here
  }
}
```

just a main method

(we're gonna put code in it in the next step)

3 In your tester, make an object and access the object's variables and methods

```
class DogTestDrive {
   public static void main (String[] args) {
     Dog d = new Dog();
     d.size = 40;
     d.bark();
   }
}
```

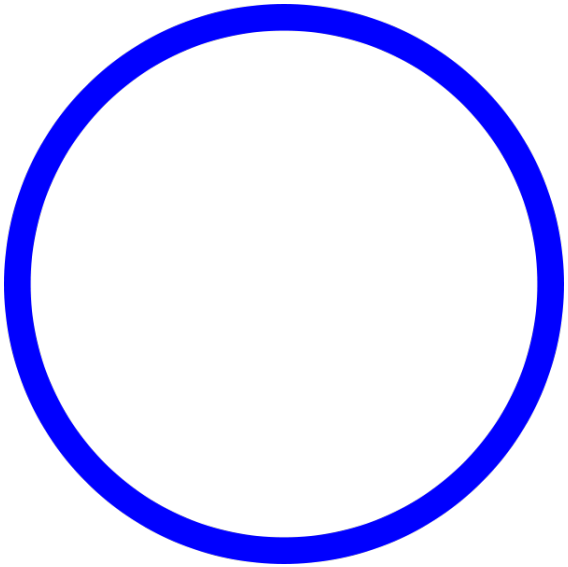
make a Dog object

use the dot operator (.) to set the size of the Dog

and to call its bark() method

dot operator

- نحوه ایجاد اشیا، دسترسی به داده های آنها و استفاده از متدها را با مثال نشان دهید.

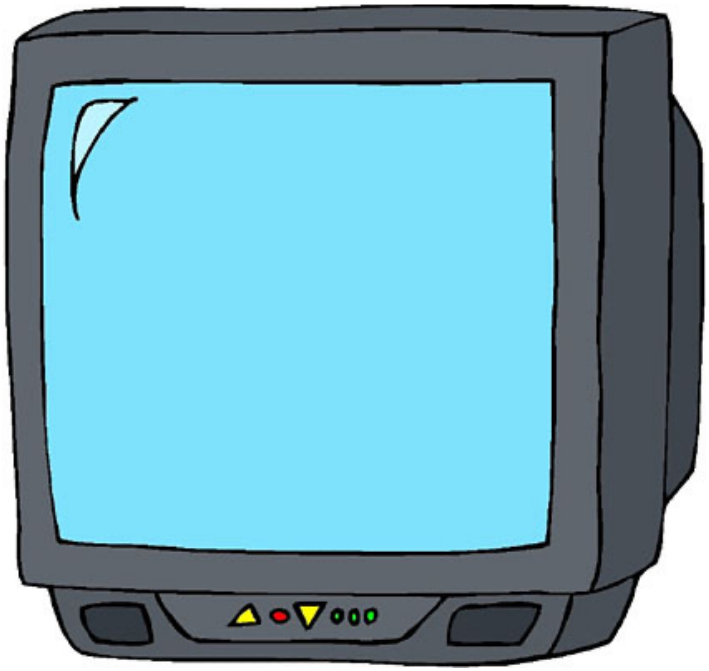


تعریف کلاسها و ایجاد اشیا-مثال

```
public class TestCircle1 {
  /** Main method */
  public static void main(String[] args) {
    // Create a circle with radius 5.0
    Circle1 myCircle = new Circle1(5.0);
    System.out.println("The area of the circle of radius “ + myCircle.radius + " is " + myCircle.getArea());

    // Create a circle with radius 1
    Circle1 yourCircle = new Circle1();
    System.out.println("The area of the circle of radius “  + yourCircle.radius + " is " + yourCircle.getArea());

    // Modify circle radius
    yourCircle.radius = 100;
    System.out.println("The area of the circle of radius “ + yourCircle.radius + " is " + yourCircle.getArea());
  }
}

// Define the circle class with two constructors
class Circle1 {
  double radius;

     /** Construct a circle with radius 1 */
     Circle1() {
        radius = 1.0;
      }

     /** Construct a circle with a specified radius */
    Circle1(double newRadius) {
       radius = newRadius;
     }

  /** Return the area of this circle */
    double getArea() {
    return radius * radius * Math.PI;
   }
}
```

تعریف کلاسها و ایجاد اشیا-مثال

- یک کلاس TV تعریف کنید و نحوه ایجاد اشیا از آن و دسترسی به داده ها و متدهای آن را نشان دهید.



تعریف کلاسها و ایجاد اشیا-مثال

```
public class TV {
  int channel = 1; // Default channel is 1
  int volumeLevel = 1; // Default volume level is 1
  boolean on = false; // By default TV is off

  public TV() {
  }

  public void turnOn() {
    on = true;
  }

  public void turnOff() {
    on = false;
  }

  public void setChannel(int newChannel) {
    if (on && newChannel >= 1 && newChannel <= 120)
      channel = newChannel;
  }

  public void setVolume(int newVolumeLevel) {
    if (on && newVolumeLevel >= 1 && newVolumeLevel <= 7)
      volumeLevel = newVolumeLevel;
  }
    public void channelUp() {
    if (on && channel < 120)
      channel++;
  }
    public void channelDown() {
    if (on && channel > 1)
      channel--;
  }
    public void volumeUp() {
    if (on && volumeLevel < 7)
      volumeLevel++;
  }
    public void volumeDown() {
    if (on && volumeLevel > 1)
      volumeLevel--;
  }
}
```

تعریف کلاسها و ایجاد اشیا-مثال

```
public class TestTV {
  public static void main(String[] args) {
    TV tv1 = new TV();
    tv1.turnOn();
    tv1.setChannel(30);
    tv1.setVolume(3);

    TV tv2 = new TV();
    tv2.turnOn();
    tv2.channelUp();
    tv2.channelUp();
    tv2.volumeUp();

    System.out.println("tv1's channel is " + tv1.channel
      + " and volume level is " + tv1.volumeLevel);
    System.out.println("tv2's channel is " + tv2.channel
      + " and volume level is " + tv2.volumeLevel);
  }
}
```

نگاهی عمیقتر-اشیا چگونه ایجاد می شوند؟

Dog
name

bark()
eat()
chaseCat()
wag();
fetch();

```
Dog d = new Dog();
d.bark();
```

think of this like this

Think of a Dog reference variable as a Dog remote control. You use it to get the object to do something (invoke methods).

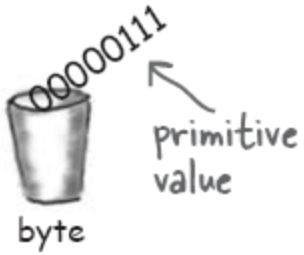
نگاهی عمیقتر-اشیا چگونه ایجاد می شوند؟

- با انواع اصلی (primitive) آشنا شدیم.
- اما در جاوا انواع ارجاعی (reference types) هم داریم.

Primitive Variable

```
byte x = 7;
```

The bits representing 7 go into the variable. (00000111).

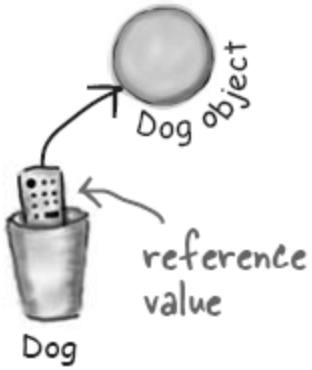


Reference Variable

```
Dog myDog = new Dog();
```

The bits representing a way to get to the Dog object go into the variable.

The Dog object itself does not go into the variable!



نگاهی عمیقتر-مراحل اعلان،ایجاد و انتساب شیئ جدید

```
Dog myDog = new Dog();
```
1 — 3 — 2

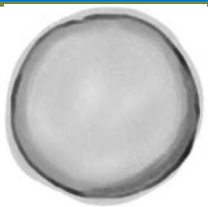
1 Declare a reference variable

```
Dog myDog = new Dog();
```

دستور نشان داده شده در کادرقرمزبه JVM می گوید تا فضایی را برای متغیرارجاعی اختصاص دهد و نام این متغیر را myDog بگذارد. این متغیرارجاعی تا ابد از نوع Dog خواهد بود. یعنی، ریموت کنترلی که دکمه هایی برای کنترل یک سگ دارد، اما نمی تواند یک گربه، یک دایره یا یک ربات را کنترل کند.

2 Create an object

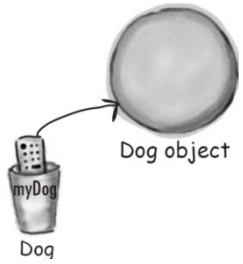
```
Dog myDog = new Dog();
```



دستور نشان داده شده در کادرقرمزبه JVM می گوید تا فضایی را برای شیئ جدید Dog در درون heap اختصاص دهد.

3 Link the object and the reference

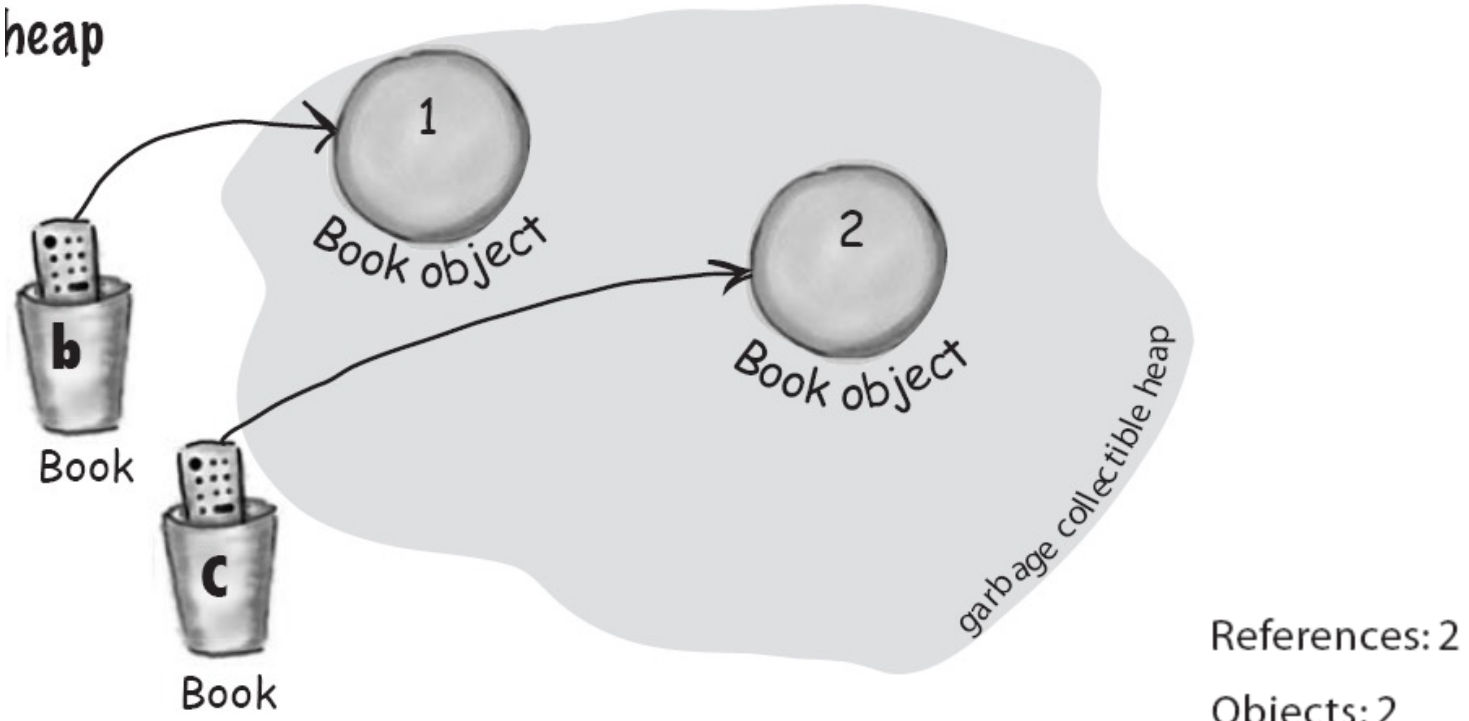
```
Dog myDog = new Dog();
```



شیئ جدید را به متغیرارجاعی myDog مرتبط می کند. به بیان دیگر، ریموت کنترل را برای هدایت شیئ جدید از نوع Dog فعال می کند.

نگاهی عمیقتر-مثال بیشتر

```
Book b = new Book();
Book c = new Book();
```

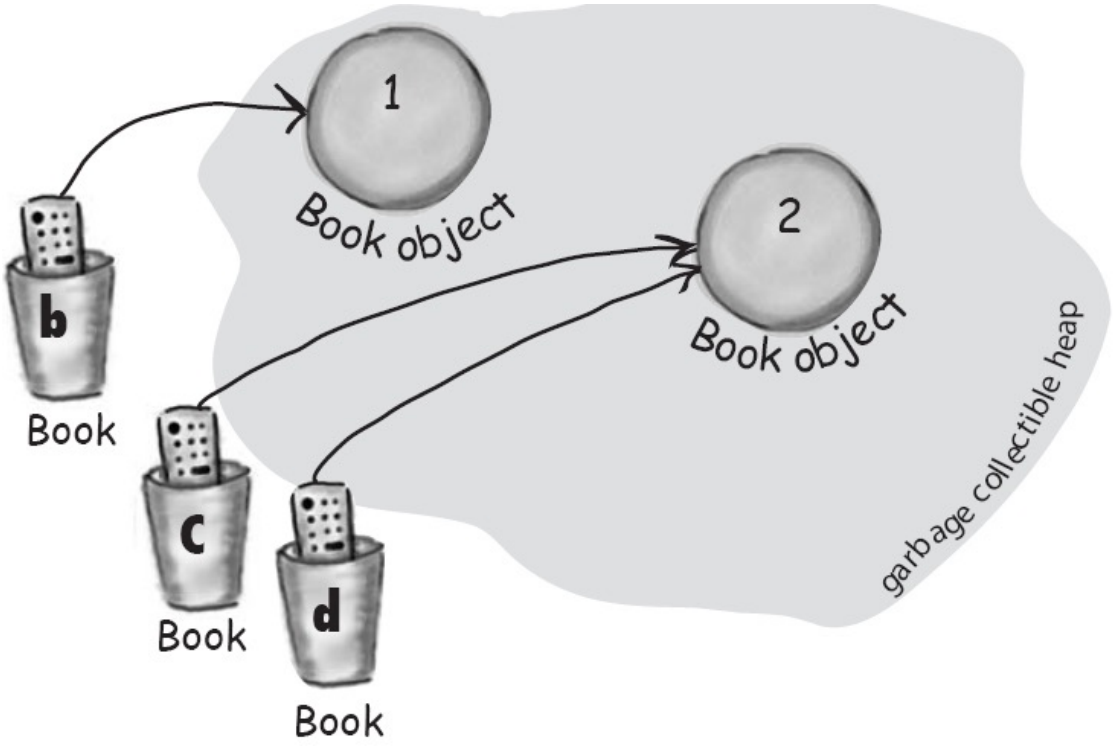


References: 2

Objects: 2

نگاهی عمیقتر-مثال بیشتر

```
Book d = c;
```

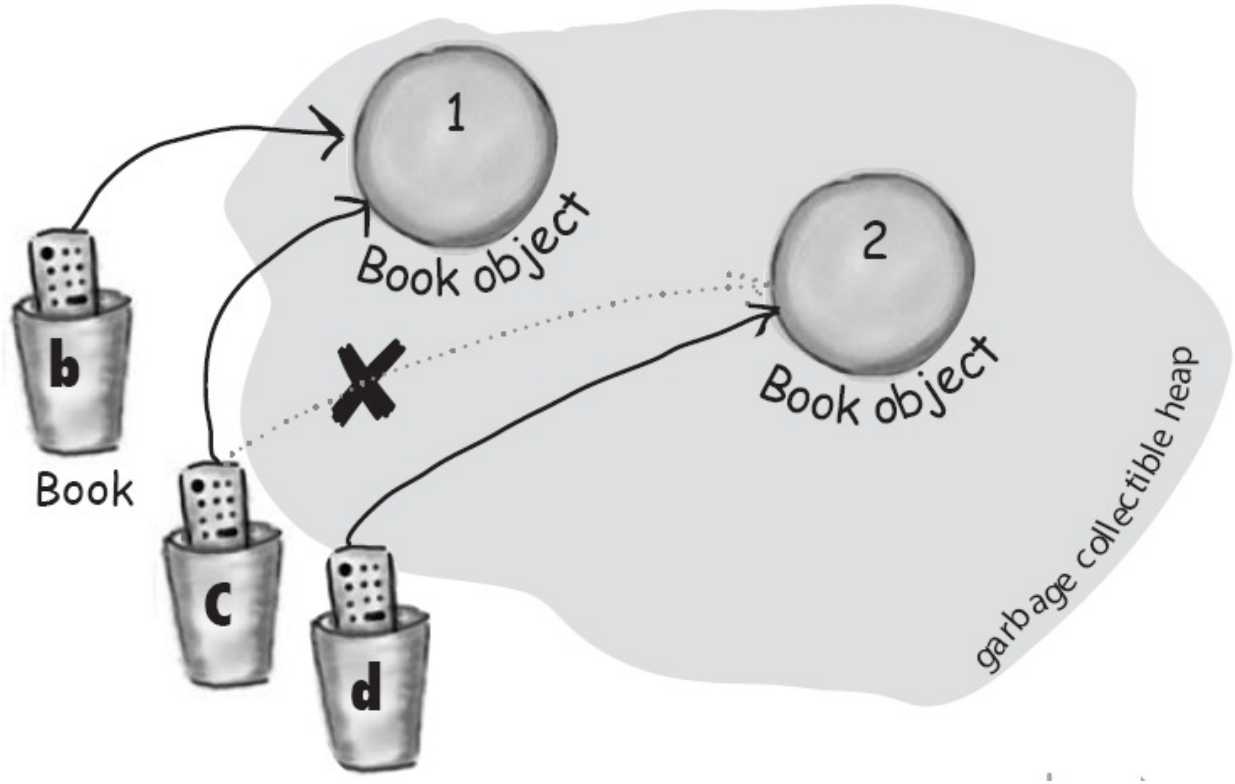


References: 3

Objects: 2

نگاهی عمیقتر-مثال بیشتر

```
c = b;
```



References: 3

Objects: 2

سازنده ها (Constructors)

```
public class TV {
  int channel = 1; // Default channel is 1
  int volumeLevel = 1; // Default volume level is 1
  boolean on = false; // By default TV is off

  public TV() {
  }

  public void turnOn() {
    on = true;
  }

  public void turnOff() {
    on = false;
  }

  public void setChannel(int newChannel) {
    if (on && newChannel >= 1 && newChannel <= 120)
      channel = newChannel;
  }

  public void setVolume(int newVolumeLevel) {
    if (on && newVolumeLevel >= 1 && newVolumeLevel <= 7)
      volumeLevel = newVolumeLevel;
  }
    public void channelUp() {
    if (on && channel < 120)
      channel++;
  }
    public void channelDown() {
    if (on && channel > 1)
      channel--;
  }
    public void volumeUp() {
    if (on && volumeLevel < 7)
      volumeLevel++;
  }
    public void volumeDown() {
    if (on && volumeLevel > 1)
      volumeLevel--;
  }
}
```

سازنده ها(Constructors)

سازنده ها نوع خاصی از متدها هستند که برای ایجاد اشیا فراخوانی می شوند.

```
Circle() {
}

Circle(double newRadius) {
  radius = newRadius;
}
```

سازنده ها (Constructors)

- یک سازنده بدون پارامتر *no-arg constructor* نامیده می شود.
- سازنده ها باید هم نام کلاس خود باشند.
- سازنده ها مقدار برگشتی ندارند-حتی void نیستند.
- سازنده ها با استفاده از عملگر new فراخوانی می شوند.
- سازنده ها نقش مقداردهی اولیه اشیای ساخته شده از آن کلاس را دارند.

ایجاد اشیا با استفاده از سازنده ها

```
new ClassName();
```

مثال:

```
new Circle();

new Circle(5.0);
```

سازنده پیش فرض

- یک کلاس می تواند بدون سازنده ها تعریف شود.

- در اینصورت، یک سازنده بدون ارگومان (no-arg) با بدنه خالی به طور ضمنی در داخل کلاس تعریف می شود.

- این سازنده که به آن سازنده پیش فرض می گویند، در صورت عدم تعریف سازنده توسط کاربر به طور خودکار توسط جاوا در داخل کلاس تعریف می شود.

اعلان متغیرهای ارجاع به شیئ

- برای ارجاع به یک شیئ، آن را به یک متغیر از نوع ارجاعی (reference) منتسب کنید.

- برای اعلان یک متغیر از نوع ارجاعی، از قاعده نحوی زیر استفاده کنید:

```
ClassName objectRefVar;
```

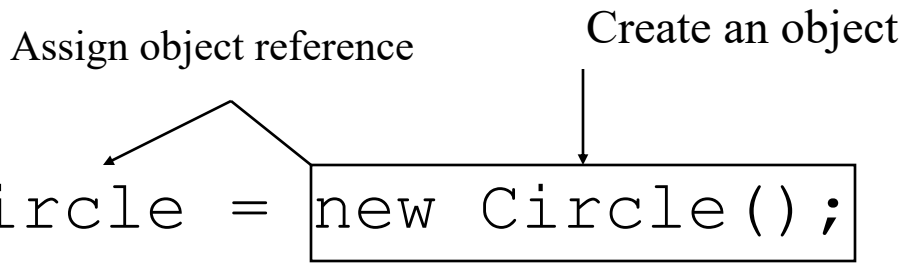
مثال:

```
Circle myCircle;
```

اعلان و ایجاد اشیا در یک مرحله

```
ClassName objectRefVar = new ClassName();
```

مثال



دسترسی به اشیا

- ارجاع (دسترسی) به داده های درون شیئ

```
objectRefVar.data
```

مثال, `myCircle.radius`

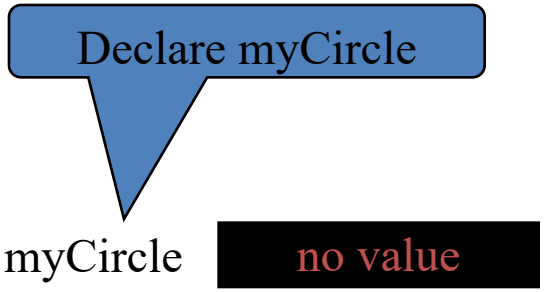
- فراخوانی متد درون یک شیئ

```
objectRefVar.methodName(arguments)
```

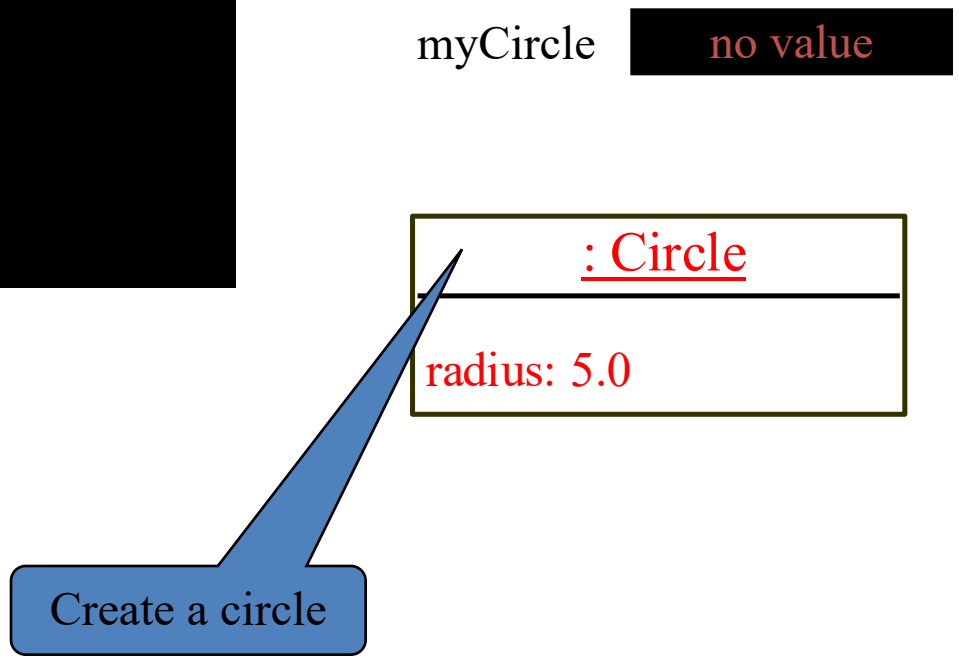
مثال, `myCircle.getArea()`

ردگیری کد

```
Circle myCircle = new Circle(5.0);

SCircle yourCircle = new Circle();

yourCircle.radius = 100;
```

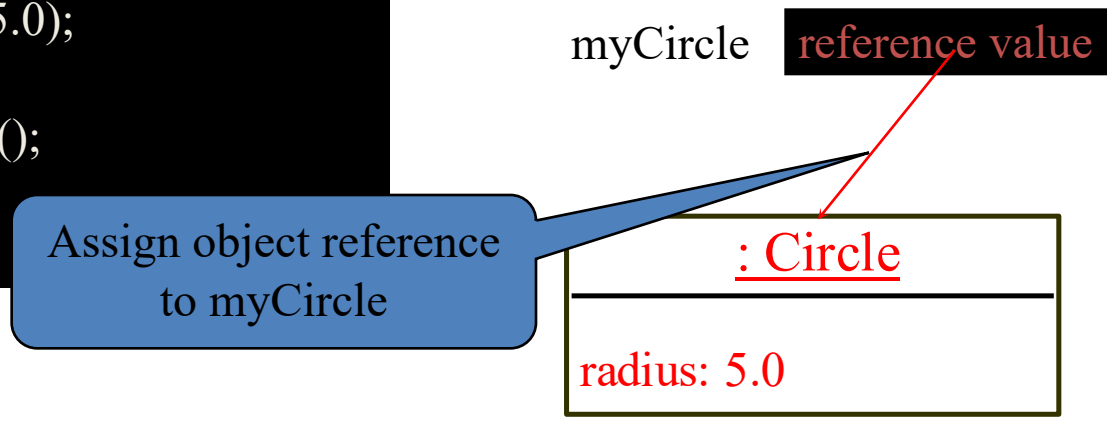


```
Circle myCircle = new Circle(5.0);

Circle yourCircle = new Circle();

yourCircle.radius = 100;
```



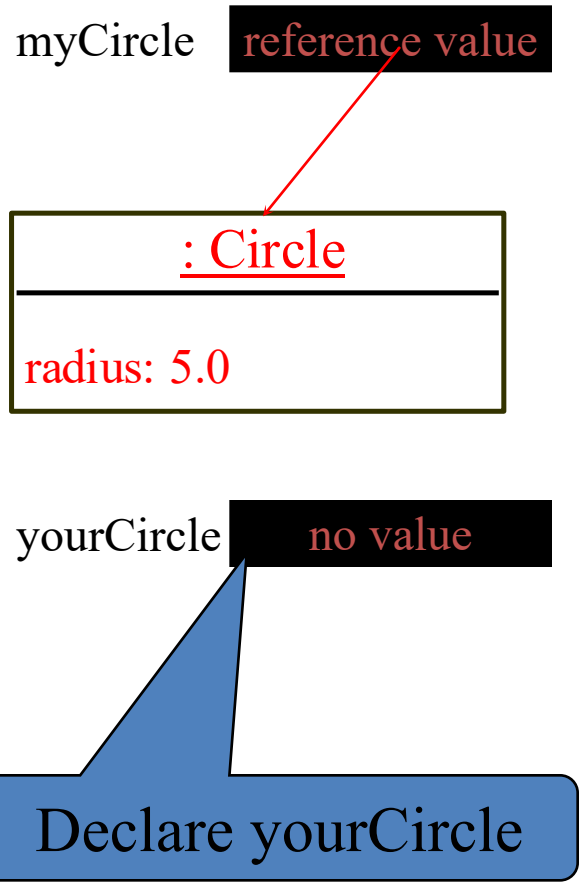
ردگیری کد

```
Circle myCircle = new Circle(5.0);

Circle yourCircle = new Circle();

yourCircle.radius = 100;
```

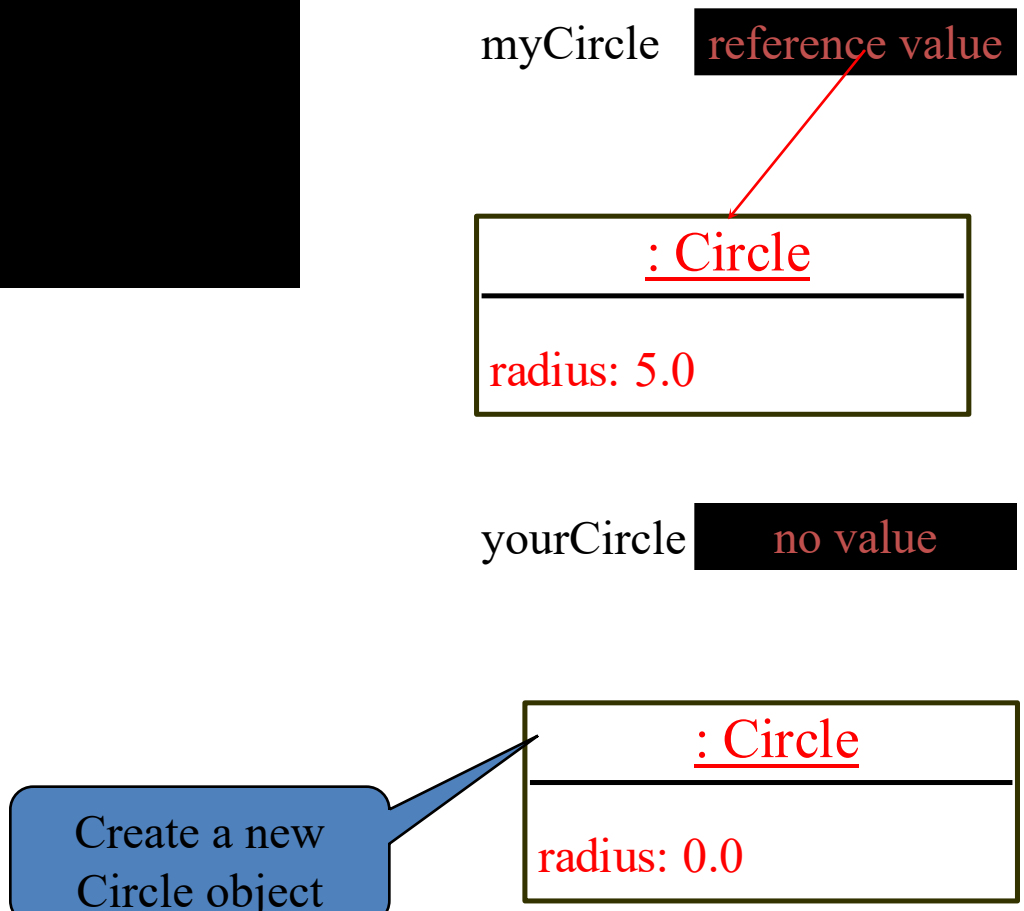


ردگیری کد

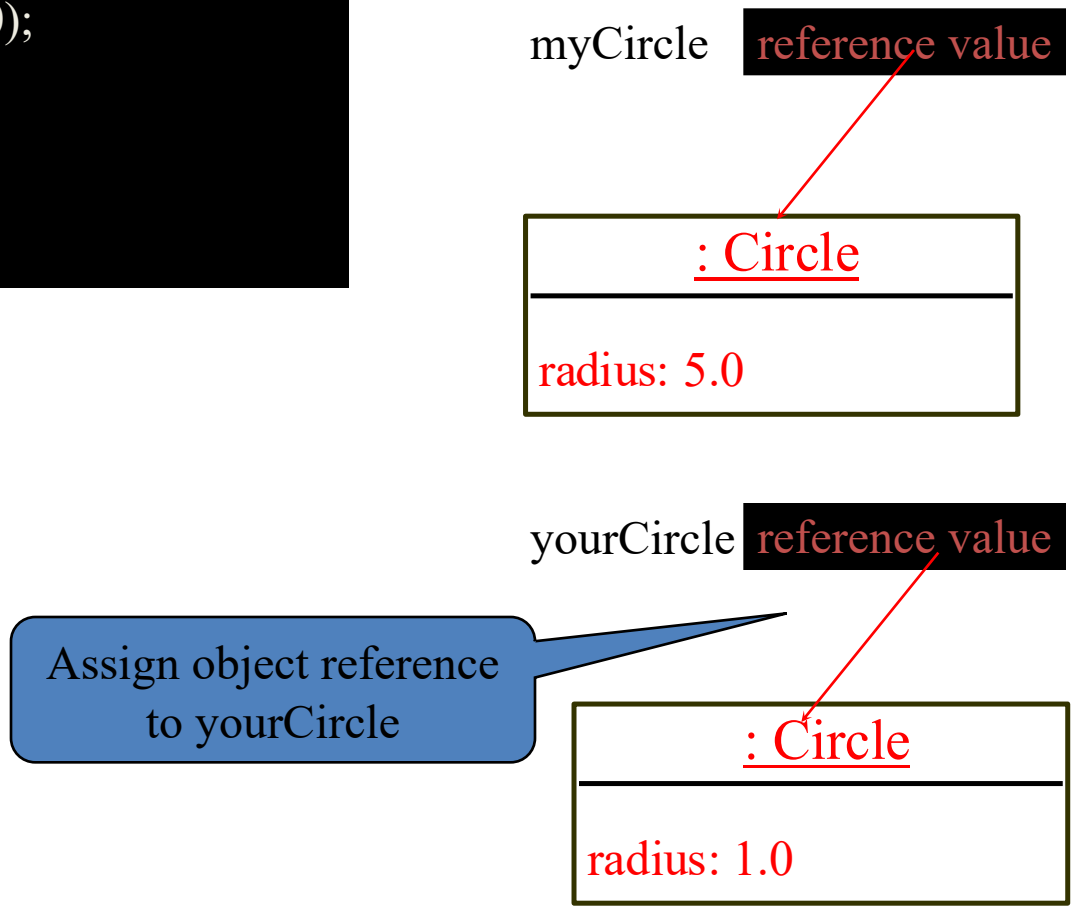
```
Circle myCircle = new Circle(5.0);

Circle yourCircle = new Circle();

yourCircle.radius = 100;
```



```
Circle myCircle = new Circle(5.0);

Circle yourCircle = new Circle();

yourCircle.radius = 100;
```

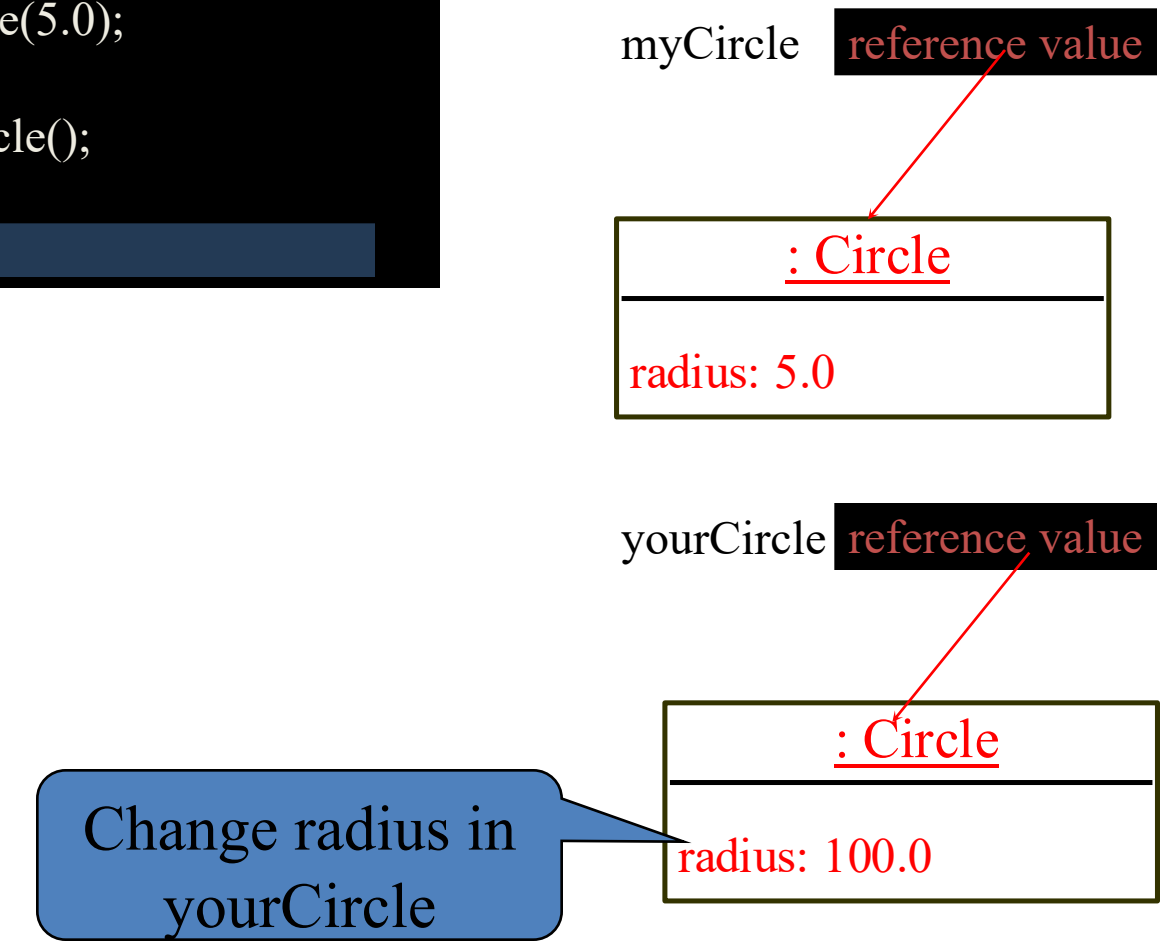


```
Circle myCircle = new Circle(5.0);

Circle yourCircle = new Circle();

yourCircle.radius = 100;
```



ردگیری کد

```
Circle myCircle = new Circle(5.0);

Circle yourCircle = new Circle();

yourCircle.radius = 100;
```



فیلدهای داده ای ارجاعی

- برخی از فیلدهای داده ای یک کلاس می توانند از نوع ارجاعی تعریف شوند.
- برای مثال، کلاس Student زیر دارای فیلد داده ای name از نوع String است.
- نوع String و آرایه از نوع ارجاعی هستند، که مفصلاً توضیح داده خواهند شد.

```
public class Student {
  String name; // name has default value null
  int age; // age has default value 0
  boolean isScienceMajor; // isScienceMajor has default value false
  char gender; // c has default value '\u0000'
}
```

مقدار null

- اگر یک فیلد داده ای از نوع ارجاعی، به شیئی اشاره (ارجاع) نداشته باشد، محتوی مقدار null خواهد بود.

```
Circle mycrircle;
```

مقدار پیش فرض برای فیلدهای داده ای

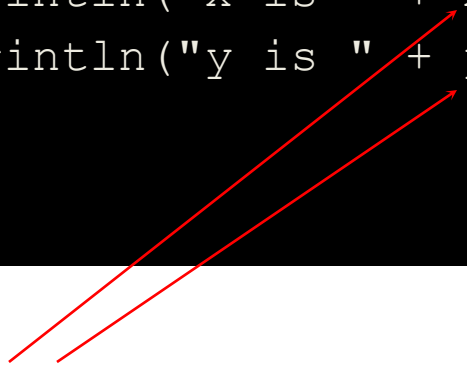
- مقدار پیش فرض برای فیلد داده ای از نوع ارجاعی null، از نوع عددی 0 ، از نوع بولین false و از نوع کاراکتری ‘\u0000’ می باشد.
- با اینحال، جاوا برای یک متغیرمحلی درون یک متد، مقدار پیش فرض در نظر نمی گیرد.

```
public class Test {
  public static void main(String[] args) {
    Student student = new Student();
    System.out.println("name? " + student.name);
    System.out.println("age? " + student.age);
    System.out.println("isScienceMajor? " + student.isScienceMajor);
    System.out.println("gender? " + student.gender);
  }
}
```

مثال

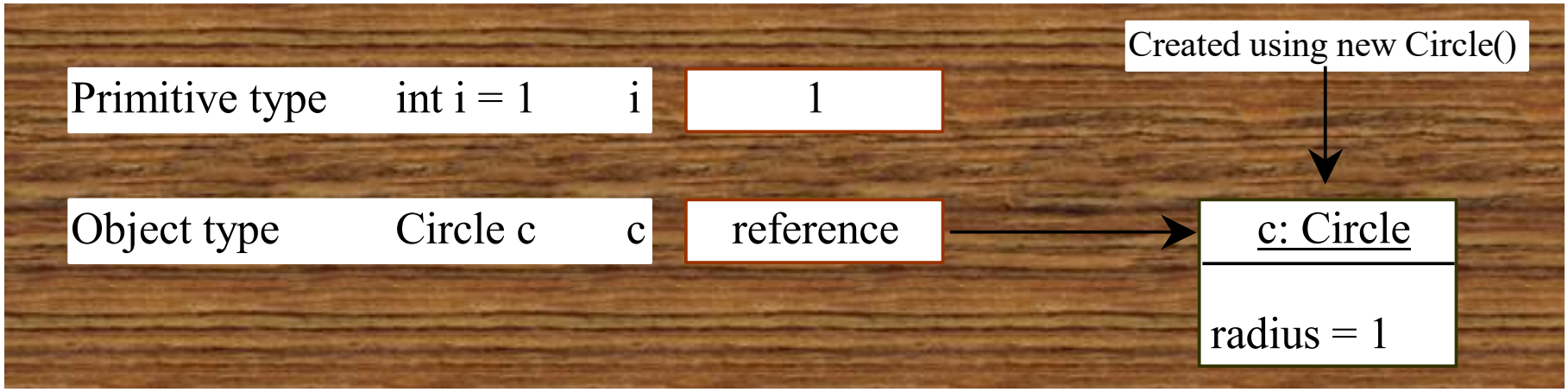
- جاوا برای یک متغیرمحلی درون یک متد، مقدار پیش فرض در نظر نمی گیرد.

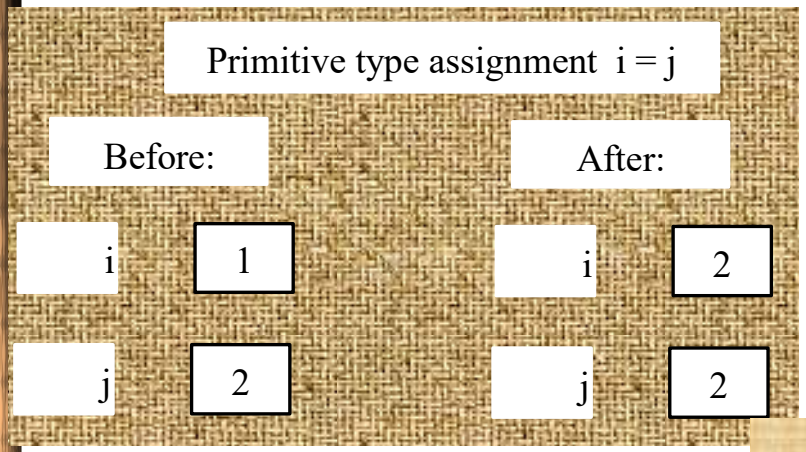
```
public class Test {
  public static void main(String[] args) {
    int x; // x has no default value
    String y; // y has no default value
    System.out.println("x is " + x);
    System.out.println("y is " + y);
  }
}
```

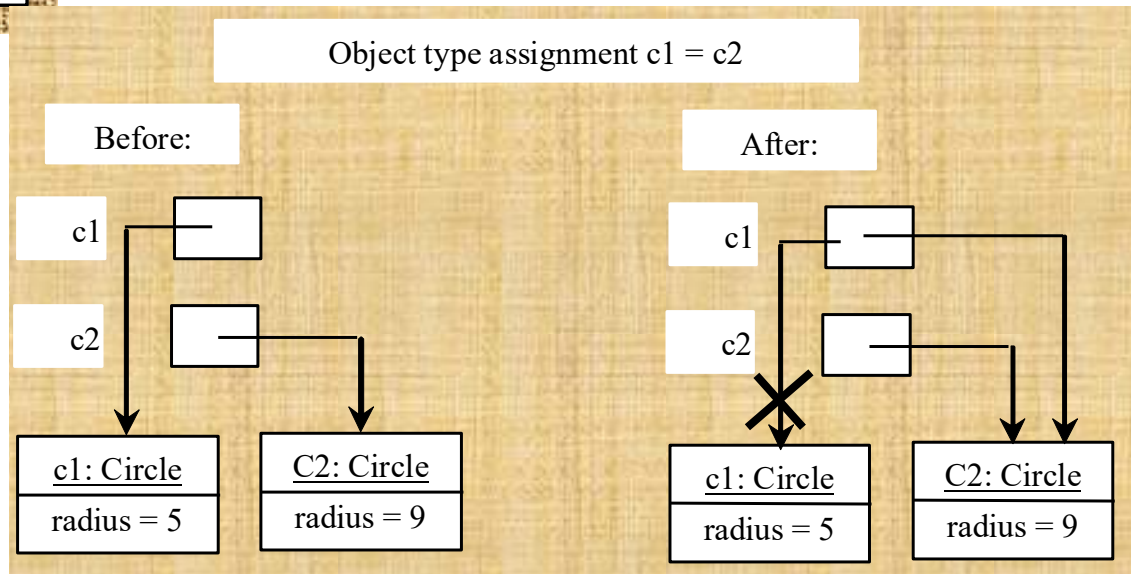


خطای زمان کامپایل: متغیرها مقداردهی اولیه نشده اند

تفاوتهای میان انواع داده ای اصلی (اولیه)و انواع ارجاعی




Primitive type assignment i = j
Before:
After:
i
1
j
2
i
2
j
2


Object type assignment c1 = c2
Before:
After:
c1
c2
c1: Circle
radius = 5
C2: Circle
radius = 9
c1
c2
c1: Circle
radius = 5
C2: Circle
radius = 9

جمع آوری زباله (garbage collection)

- همان طور که در شکل اسلاید قبل دیدیم، پس از دستور انتسابی c1 = c2، c1 به همان شیئی اشاره می کند که توسط c2 مورد ارجاع قرار گرفته است.
- در اینحالت، شیئی که قبلاً توسط c1 مورد ارجاع قرار می گرفت، دیگر توسط کسی مورد اشاره قرار نمی گیرد.
- به این شیئ اصطلاحاً زباله گفته می شود. زباله در جاوا به طور خودکار توسط JVM جمع آوری می شود.

- نکته: اگر مطمئنید که یک شیئ، دیگر مورد نیاز نمی باشد، می توانیم به متغیر ارجاعی کنترل کننده آن شیئ، مقدار `null` بدهیم.

```
C1=null;
```

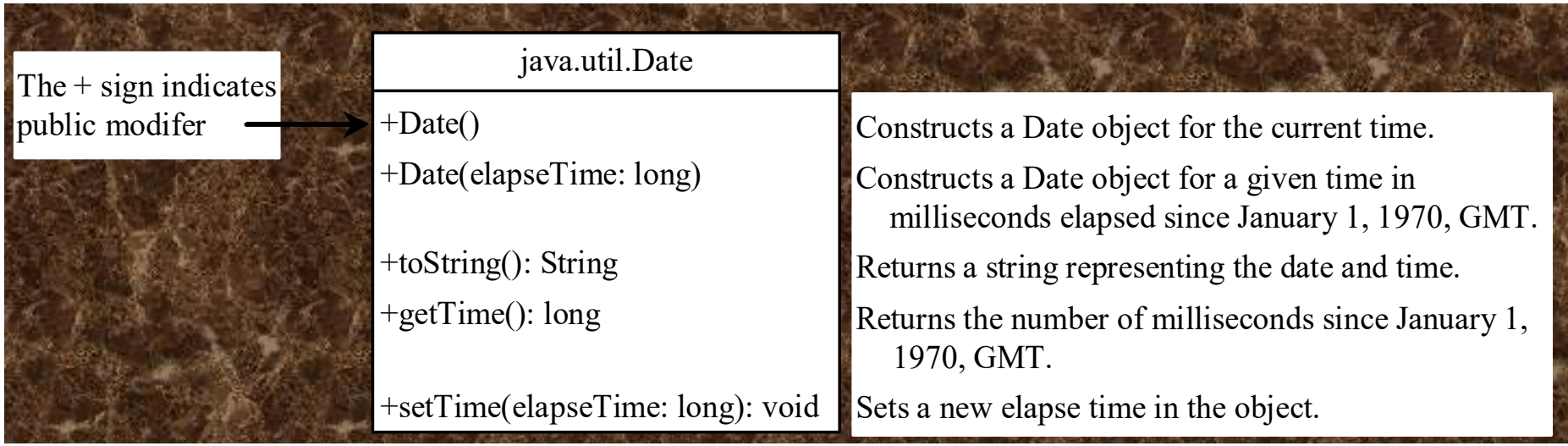
- به این ترتیب، `JVM` به طور خودکار فضای گرفته شده توسط آن شیئ را اگر اشاره کننده دیگری نداشته باشد، آزاد خواهد نمود.

- مثال کد زیر:

```
java.util.Date date = new java.util.Date();
System.out.println(date.toString());
```

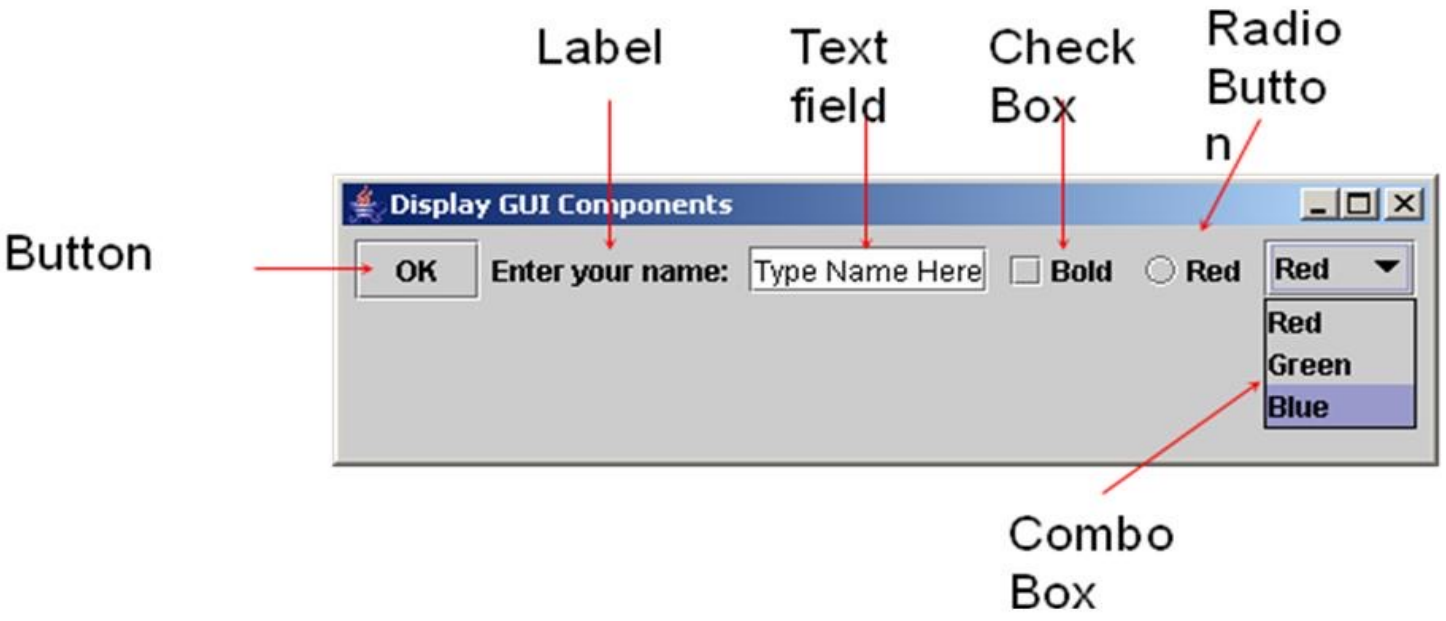
- رشته ای نظیر Wen Feb 15 09:40:19 IRST 2017 را نمایش می دهد.

- جاوا برای نمایش تاریخ و زمان از کلاس java.util.Date استفاده می کند.
- شما می توانید از کلاس Date برای ایجاد یک شیئ از تاریخ و زمان فعلی استفاده کنید و با کمک متد toString آن تاریخ و زمان را در قالب رشته برگردانید.



•هنگامی که می خواهید برنامه ای جهت ایجاد واسطهای گرافیکی کاربر بنویسید، می توانید از کلاسهای جاوا نظیر JFrame, JButton, JRadioButton, JComboBox, و JList برای تولید فریمها، دکمه ها، دکمه های رادیویی، جعبه های کمبو، لیستها و غیره استفاده کنید.

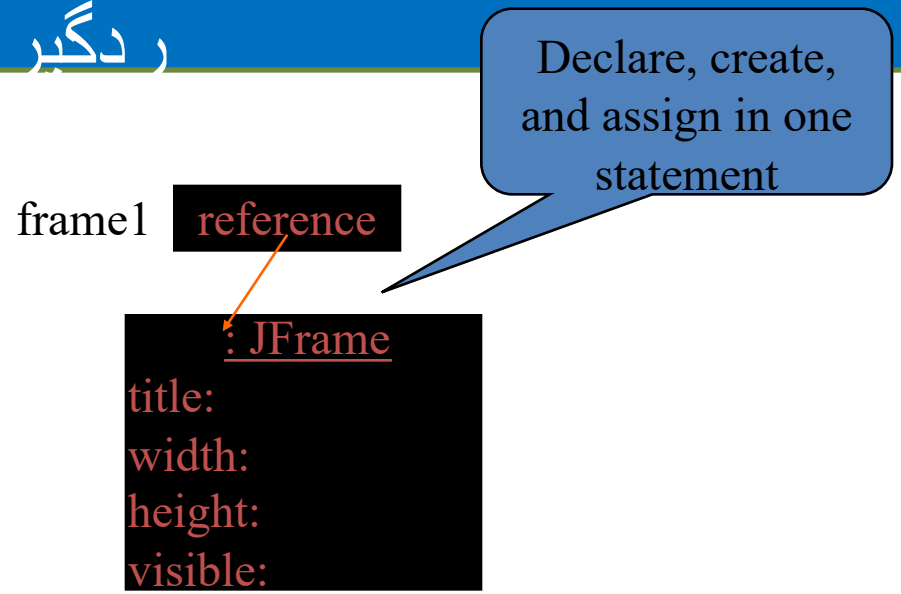
•در اینجا به کمک کلاس JFrame دو پنجره ساده ایجاد می کنیم.



```
import javax.swing.JFrame;

public class TestFrame {
  public static void main(String[] args) {
    JFrame frame1 = new JFrame();
    frame1.setTitle("Window 1");
    frame1.setSize(200, 150);
    frame1.setLocation(200, 100);
    frame1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame1.setVisible(true);

    JFrame frame2 = new JFrame();
    frame2.setTitle("Window 2");
    frame2.setSize(200, 150);
    frame2.setLocation(410, 100);
    frame2.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame2.setVisible(true);
  }
}
```

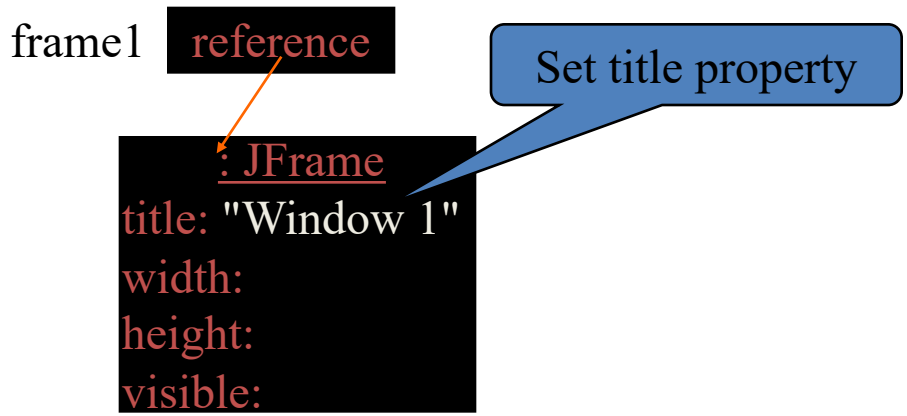
ر دگیر ی کد

```
JFrame frame1 = new JFrame();
frame1.setTitle("Window 1");
frame1.setSize(200, 150);
frame1.setVisible(true); JFrame
frame2 = new JFrame();
frame2.setTitle("Window 2");
frame2.setSize(200, 150);
frame2.setVisible(true);
```



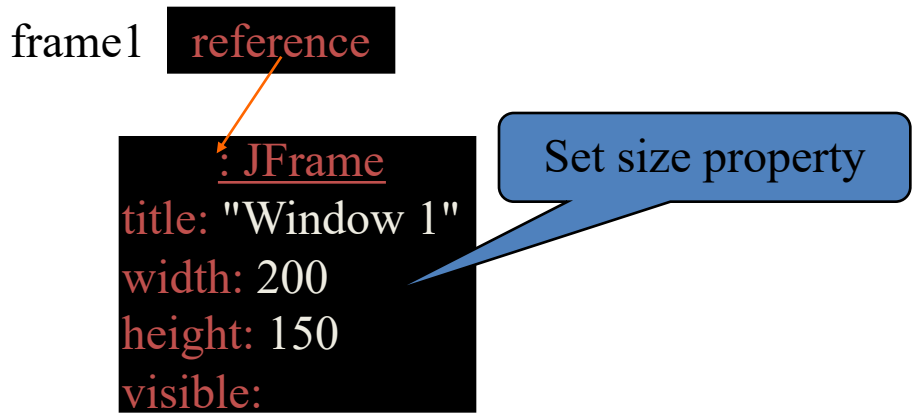
ردگیری کد

```
JFrame frame1 = new JFrame();
frame1.setTitle("Window 1");
frame1.setSize(200, 150);
frame1.setVisible(true); JFrame
frame2 = new JFrame();
frame2.setTitle("Window 2");
frame2.setSize(200, 150);
frame2.setVisible(true);
```



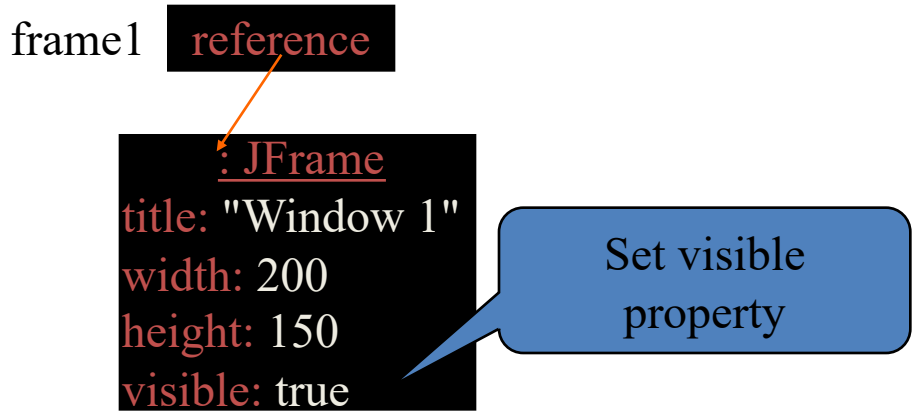
ردگیری کد

```
JFrame frame1 = new JFrame();
frame1.setTitle("Window 1");
frame1.setSize(200, 150);
frame1.setVisible(true);
JFrame frame2 = new JFrame();
frame2.setTitle("Window 2");
frame2.setSize(200, 150);
frame2.setVisible(true);
```



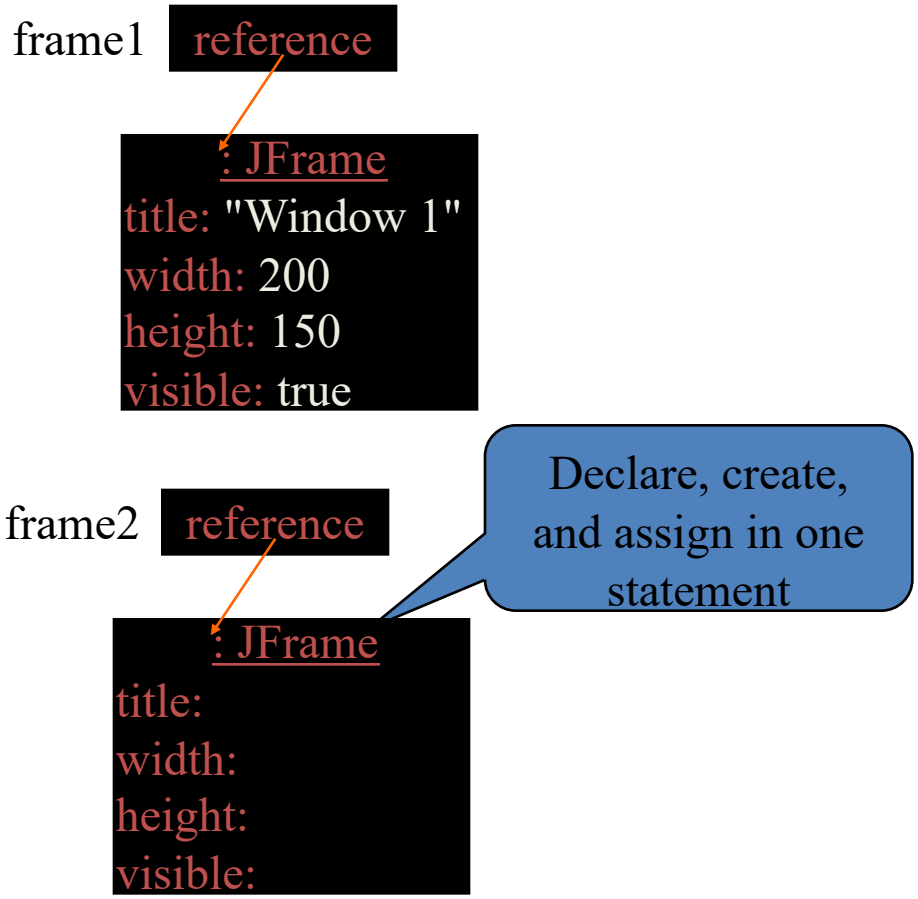
ردگیری کد

```
JFrame frame1 = new JFrame();
frame1.setTitle("Window 1");
frame1.setSize(200, 150);
frame1.setVisible(true);
JFrame frame2 = new JFrame();
frame2.setTitle("Window 2");
frame2.setSize(200, 150);
frame2.setVisible(true);
```



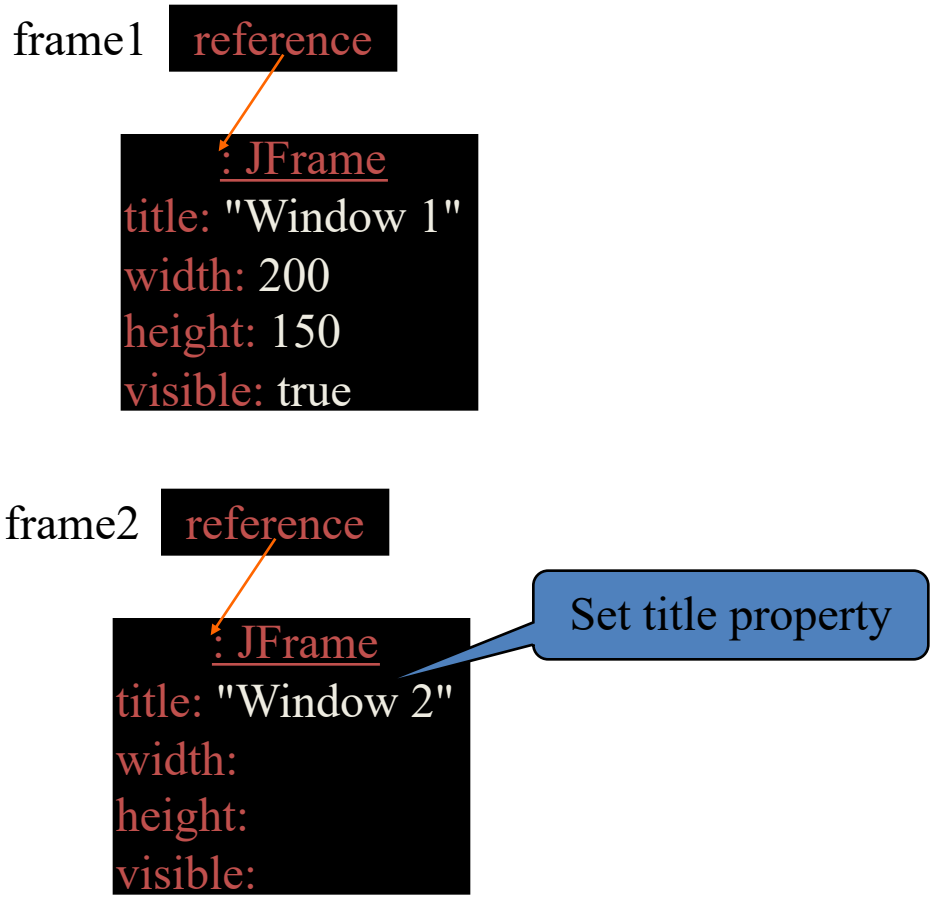
ردگیری کد

```
JFrame frame1 = new JFrame();
frame1.setTitle("Window 1");
frame1.setSize(200, 150);
frame1.setVisible(true);
JFrame frame2 = new JFrame();
frame2.setTitle("Window 2");
frame2.setSize(200, 150);
frame2.setVisible(true);
```



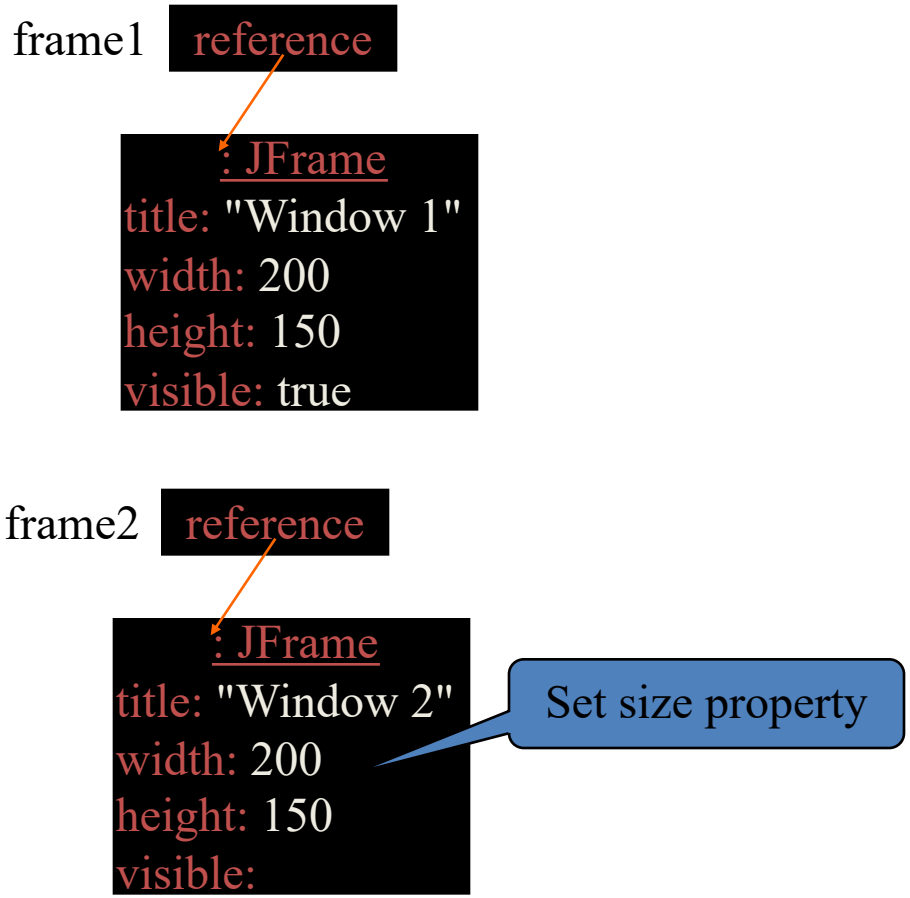
ردگیری کد

```
JFrame frame1 = new JFrame();
frame1.setTitle("Window 1");
frame1.setSize(200, 150);
frame1.setVisible(true);
JFrame frame2 = new JFrame();
frame2.setTitle("Window 2");
frame2.setSize(200, 150);
frame2.setVisible(true);
```



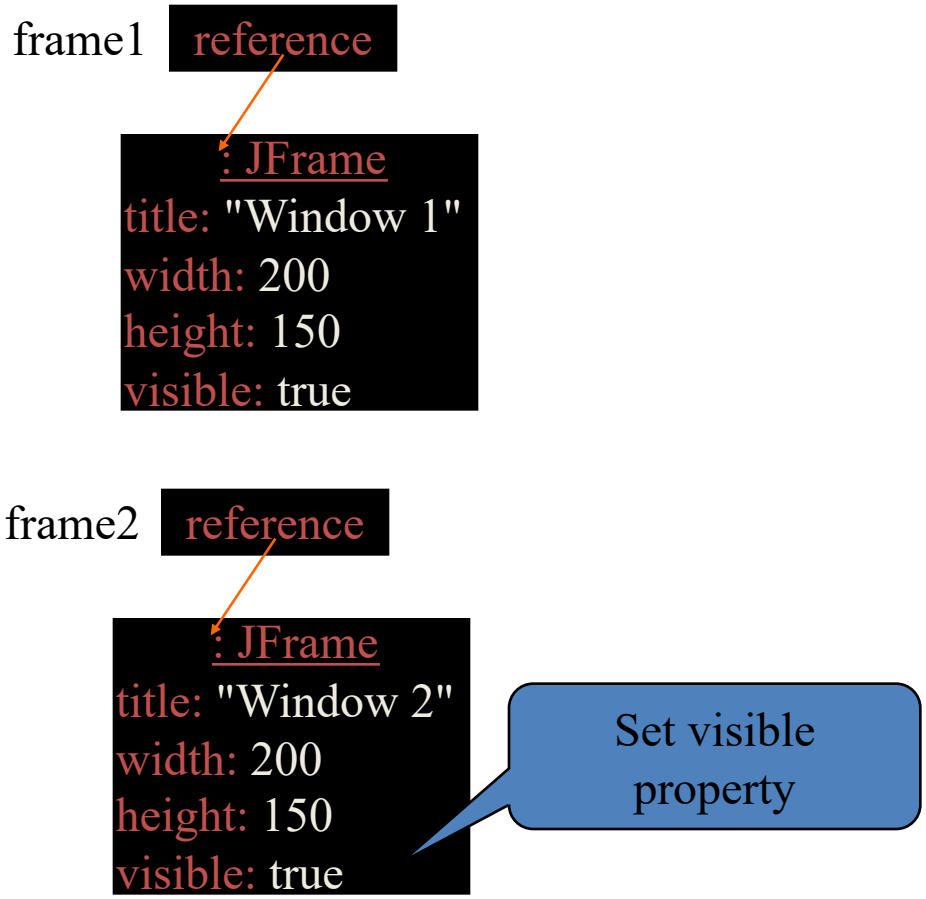
ردگیری کد

```
JFrame frame1 = new JFrame();
frame1.setTitle("Window 1");
frame1.setSize(200, 150);
frame1.setVisible(true);
JFrame frame2 = new JFrame();
frame2.setTitle("Window 2");
frame2.setSize(200, 150);
frame2.setVisible(true);
```



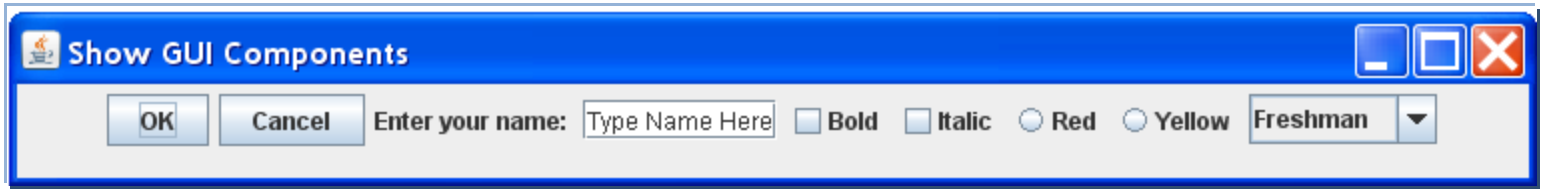
ردگیری کد

```
JFrame frame1 = new JFrame();
frame1.setTitle("Window 1");
frame1.setSize(200, 150);
frame1.setVisible(true);
JFrame frame2 = new JFrame();
frame2.setTitle("Window 2");
frame2.setSize(200, 150);
frame2.setVisible(true);
```



اضافه نمودن مولفه های گرافیکی به پنجره

- این مولفه های گرافیکی با استفاده از کلاسهای کتابخانه جاوا به سادگی ایجاد می شوند.
- در اسلاید بعدی با یک مثال نحوه ایجاد این مولفه ها نشان داده شده است.



اضافه نمودن مولفه های گرافیکی به پنجره

```
import javax.swing.*;

public class GUIComponents {
  public static void main(String[] args) {
    // Create a button with text OK
    JButton jbtOK = new JButton("OK");

    // Create a button with text Cancel
    JButton jbtCancel = new JButton("Cancel");

    // Create a label with text "Enter your name: "
    JLabel jlblName = new JLabel("Enter your name: ");

    // Create a text field with text "Type Name Here"
    JTextField jtfName = new JTextField("Type Name Here");

    // Create a check box with text bold
    JCheckBox jchkBold = new JCheckBox("Bold");

    // Create a check box with text italic
    JCheckBox jchkItalic = new JCheckBox("Italic");

    // Create a radio button with text red
    JRadioButton jrbRed = new JRadioButton("Red");

    // Create a radio button with text yellow
    JRadioButton jrbYellow = new JRadioButton("Yellow");

    // Create a combo box with several choices
    JComboBox jcboColor = new JComboBox(new String[]{"Freshman",
      "Sophomore", "Junior", "Senior"});

    // Create a panel to group components
    JPanel panel = new JPanel();
    panel.add(jbtOK); // Add the OK button to the panel
    panel.add(jbtCancel); // Add the Cancel button to the panel
    panel.add(jlblName); // Add the label to the panel
    panel.add(jtfName); // Add the text field to the panel
    panel.add(jchkBold); // Add the check box to the panel
    panel.add(jchkItalic); // Add the check box to the panel
    panel.add(jrbRed); // Add the radio button to the panel
    panel.add(jrbYellow); // Add the radio button to the panel
    panel.add(jcboColor); // Add the combo box to the panel

    JFrame frame = new JFrame(); // Create a frame
    frame.add(panel); // Add the panel to the frame
    frame.setTitle("Show GUI Components");
    frame.setSize(450, 100);
    frame.setLocation(200, 100);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
```

متغیر ها و متدهای نمونه ای

- متغیر های نمونه (Instance variables) مختص به یک شیئ خاص هستند که از یک کلاسِ حاوی آن فیلدها ایجاد شده اند.

- متدهای نمونه (Instance methods) توسط متغیر ارجاعی (ریموت کنترل) به یک شیئ فراخوانی می شوند.

- متغیرهای ایستا یا استاتیک از طرف همه نمونه ها (اشیای) یک کلاس به اشتراک گذاشته می شوند.
- متدهای ایستا یا استاتیک به یک شیئ خاص وابسته نیستند.
- ثابتهای استاتیک متغیرهای final هستند که از جانب همه نمونه های کلاس به اشتراک گذاشته شده اند.
- برای مشخص کردن استاتیک بودن متغیر یا متد از کلمه کلیدی static استفاده کنید.

مثال-Static Variables, Constants, and Methods

- برنامه ای بنویسید و در آن نقش متغیرهای نمونه و متغیرهای کلاس را به همراه نحوه استفاده از آنها نشان دهید.
- این مثال، به یک متغیر کلاسی numberOfObjects یک واحد اضافه می کند تا تعداد اشیای ساخته شده از کلاس Circle را حساب کند.

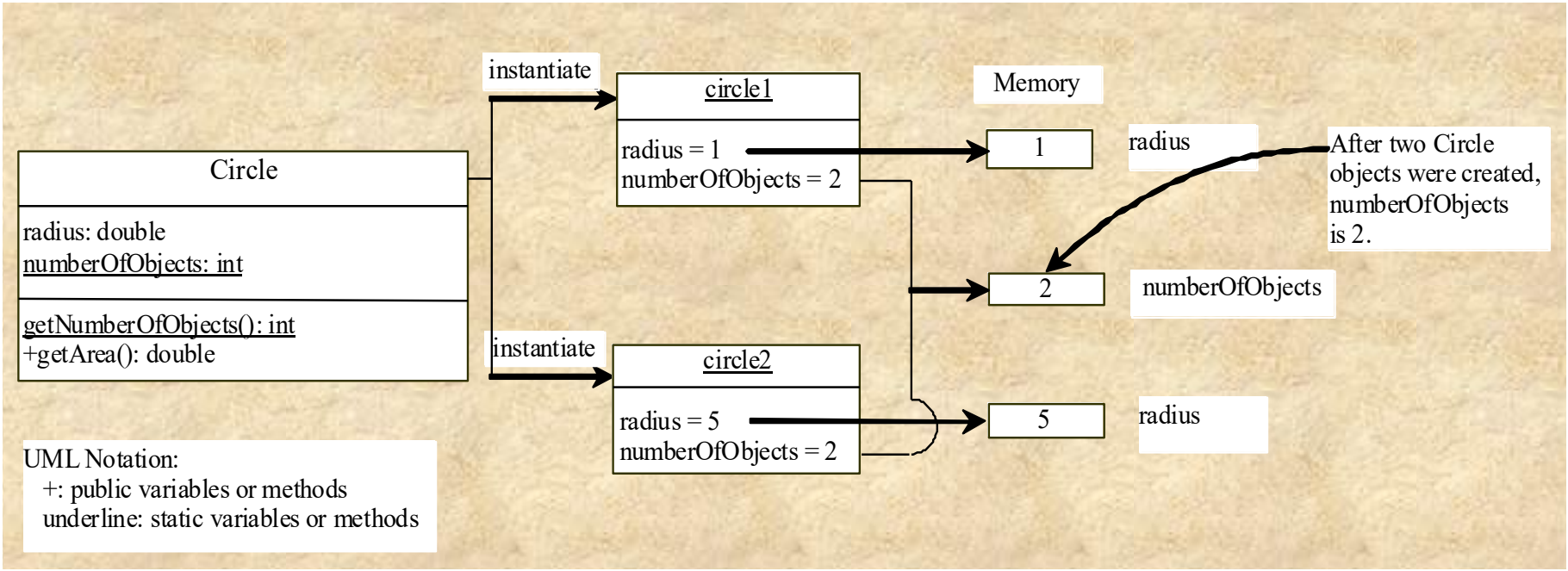
مثال-Static Variables, Constants, and Methods

```
public class Circle2 {
  /** The radius of the circle */
  double radius;

  /** The number of the objects created */
  static int numberOfObjects = 0;

  /** Construct a circle with radius 1 */
  Circle2() {
    radius = 1.0;
    numberOfObjects++;
  }

  /** Construct a circle with a specified radius */
  Circle2(double newRadius) {
    radius = newRadius;
    numberOfObjects++;
  }

  /** Return numberOfObjects */
  static int getNumberOfObjects() {
    return numberOfObjects;
  }

  /** Return the area of this circle */
  double getArea() {
    return radius * radius * Math.PI;
  }
}
```

Static Variables, Constants, and Methods-مثال

```
public class TestCircle2 {
  /** Main method */
  public static void main(String[] args) {
    System.out.println("Before creating objects");
    System.out.println("The number of Circle objects is " +  Circle2.numberOfObjects);

    // Create c1
    Circle2 c1 = new Circle2();

    // Display c1 BEFORE c2 is created
    System.out.println("\nAfter creating c1");
    System.out.println("c1: radius (" + c1.radius +   ") and number of Circle objects (" + c1.numberOfObjects + ")");

    // Create c2
    Circle2 c2 = new Circle2(5);

    // Modify c1
    c1.radius = 9;

    // Display c1 and c2 AFTER c2 was created
    System.out.println("\nAfter creating c2 and modifying c1");
    System.out.println("c1: radius (" + c1.radius + ") and number of Circle objects (" + c1.numberOfObjects + ")");
    System.out.println("c2: radius (" + c2.radius +  ") and number of Circle objects (" +  c2.numberOfObjects + ")");
  }
}
```

نحوه نمایش متغیرهای نمونه و ایستا در حافظه



•سطح دسترسی پیش فرض: به طور پیش فرض، کلاس، متغیر یا متدها می توانند توسط هرکلاس در پکیج موردنظر مورد دسترسی قرار بگیرند.

- `Public`
  - کلاس،داده یا متد برای هر کلاسی در هر پکیجی قابل مشاهده است.
- `private`
  - داده یا متد تنها از سوی کلاسی که در آن اعلان شده اند، قابل مشاهده است.

- **متدهای get و set که به آنها getter و setter هم گفته می شود، برای خواندن و تغییر اجزای private مورد استفاده قرار می گیرند.**

دقت کنید!

- یک شیئ نمی تواند به اعضای private خود دسترسی داشته باشد، اگر شیئ در کلاس دیگر ایجاد شود. (شکل (b))
- اما در شکل (a) می تواند دسترسی داشته باشد، چون شیئ در خود کلاس تعریف شده است.

```
public class Foo {
  private boolean x;

  public static void main(String[] args) {
    Foo foo = new Foo();
    System.out.println(foo.x);
    System.out.println(foo.convert());
  }

  private int convert(boolean b) {
    return x ? 1 : -1;
  }
}
```

(a) This is OK because object foo is used inside the Foo class

```
public class Test {
  public static void main(String[] args) {
    Foo foo = new Foo();
    System.out.println(foo.x);
    System.out.println(foo.convert(foo.x));
  }
}
```

(b) This is wrong because x and convert are private in Foo.

چرا باید فیلدهای داده ای private باشند؟

- برای محافظت از داده ها
- برای اینکه نگهداری از کلاس آسان باشد.

package p1;

```
public class C1 {
  public int x;
  int y;
  private int z;

  public void m1() {
  }
  void m2() {
  }
  private void m3() {
  }
}
```

```
public class C2 {
  void aMethod() {
    C1 o = new C1();
    can access o.x;
    can access o.y;
    cannot access o.z;

    can invoke o.m1();
    can invoke o.m2();
    cannot invoke o.m3();
  }
}
```

package p2;

```
public class C3 {
  void aMethod() {
    C1 o = new C1();
    can access o.x;
    cannot access o.y;
    cannot access o.z;

    can invoke o.m1();
    cannot invoke o.m2();
    cannot invoke o.m3();
  }
}
```

package p1;

```
class C1 {
  ...
}
```

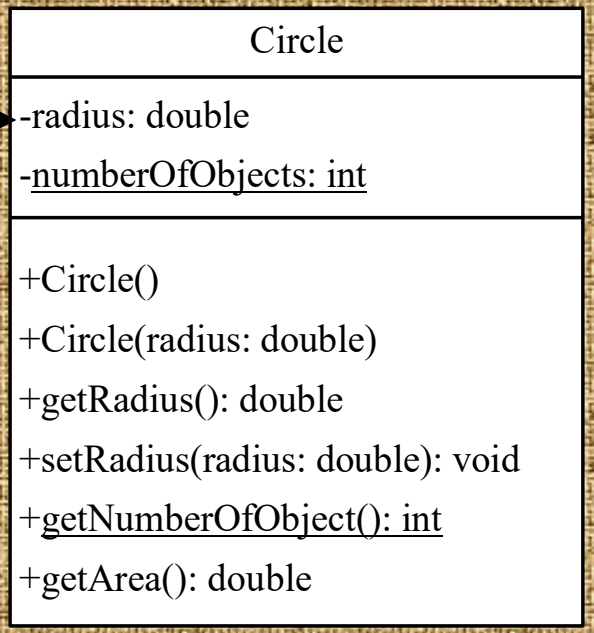
```
public class C2 {
  can access C1
}
```

package p2;

```
public class C3 {
  cannot access C1;
  can access C2;
}
```

مثالی از کپسوله بندی فیلد داده ای

The - sign indicates private modifier



The radius of this circle (default: 1.0).

The number of circle objects created.

Constructs a default circle object.

Constructs a circle object with the specified radius.

Returns the radius of this circle.

Sets a new radius for this circle.

Returns the number of circle objects created.

Returns the area of this circle.

مثالی از کپسوله بندی فیلد داده ای

```
public class Circle3 {
  /** The radius of the circle */
  private double radius = 1;

  /** The number of the objects created */
  private static int numberOfObjects = 0;

  /** Construct a circle with radius 1 */
  public Circle3() {
    numberOfObjects++;
  }

  /** Construct a circle with a specified radius */
  public Circle3(double newRadius) {
    radius = newRadius;
    numberOfObjects++;
  }

  /** Return radius */
  public double getRadius() {
    return radius;
  }

  /** Set a new radius */
  public void setRadius(double newRadius) {
    radius = (newRadius >= 0) ? newRadius : 0;
  }

  /** Return numberOfObjects */
  public static int getNumberOfObjects() {
    return numberOfObjects;
  }

  /** Return the area of this circle */
  public double getArea() {
    return radius * radius * Math.PI;
  }
}
```

مثالی از کپسوله بندی فیلد داده ای

```
public class TestCircle3 {
  /** Main method */
  public static void main(String[] args) {
    // Create a Circle with radius 5.0
    Circle3 myCircle = new Circle3(5.0);
    System.out.println("The area of the circle of radius "
      + myCircle.getRadius() + " is " + myCircle.getArea());

    // Increase myCircle's radius by 10%
    myCircle.setRadius(myCircle.getRadius() * 1.1);
    System.out.println("The area of the circle of radius "
      + myCircle.getRadius() + " is " + myCircle.getArea());
  }
}
```

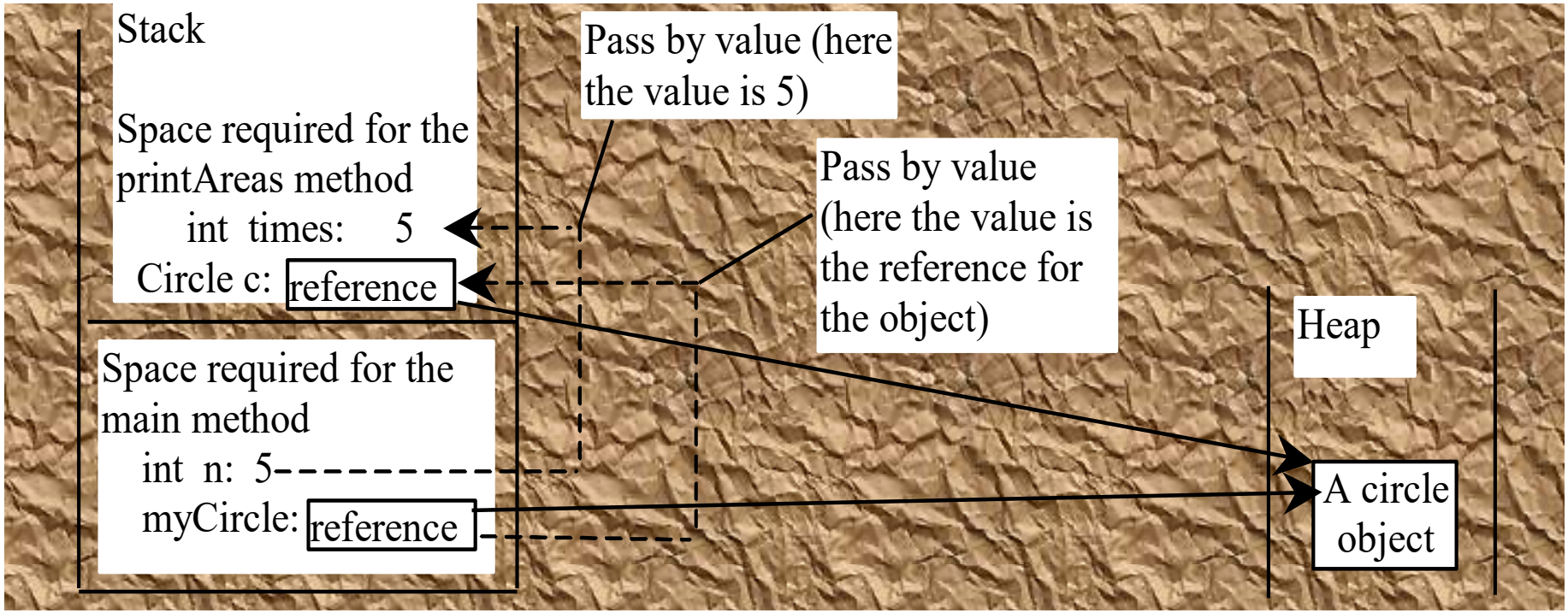
ارسال اشیا به متدها

- ارسال با مقدار برای انواع اصلی (یک کپی از مقدار ارسال می شود).
- ارسال با مقدار برای انواع ارجاعی (مقدار ارسالی ارجاع به آن شیئی است، یک کپی از ریموت کنترل شیئ!!!)

ارسال اشیا به متدها

```
public class TestPassObject {
  /** Main method */
  public static void main(String[] args) {
    // Create a Circle object with radius 1
    Circle3 myCircle = new Circle3(1);

    // Print areas for radius 1, 2, 3, 4, and 5.
    int n = 5;
    printAreas(myCircle, n);

    // See myCircle.radius and times
    System.out.println("\n" + "Radius is " + myCircle.getRadius());
    System.out.println("n is " + n);
  }

  /** Print a table of areas for radius */
  public static void printAreas(Circle3 c, int times) {
    System.out.println("Radius \t\tArea");
    while (times >= 1) {
      System.out.println(c.getRadius() + "\t\t" + c.getArea());
      c.setRadius(c.getRadius() + 1);
      times--;
    }
  }
}
```

ارسال اشیا به متدها



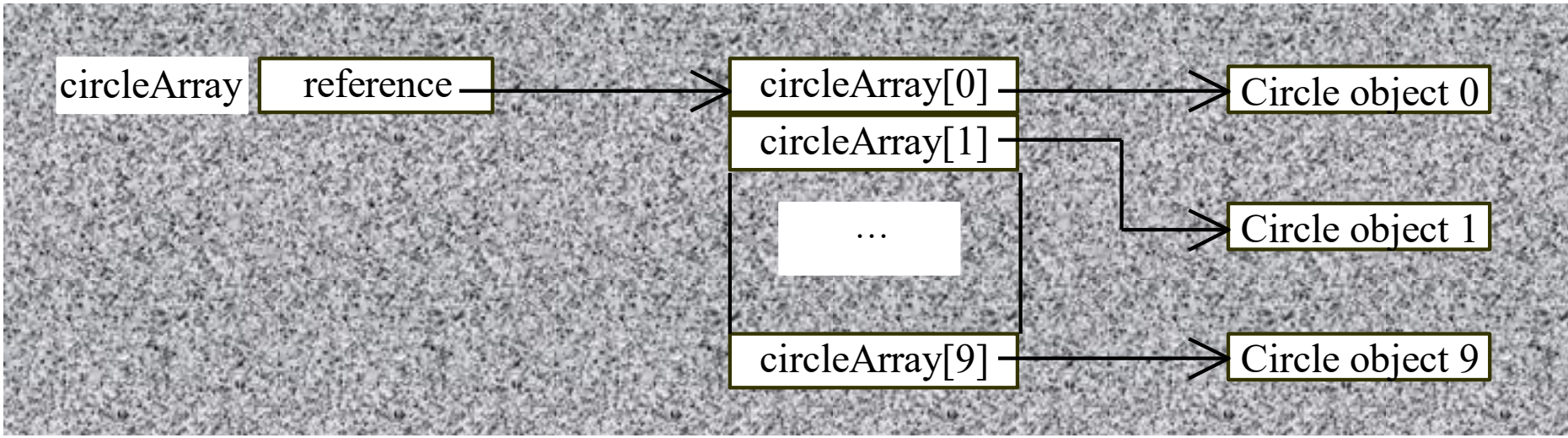
آرایه ای از اشیا

```
Circle[] circleArray = new Circle[10];
```

- آرایه ای از اشیا درواقع آرایه ای از متغیرهای ارجاعی می باشد.
- در نتیجه فراخوانی circleArray[1].getArea() شامل دو سطح ارجاع دهی می باشد که در اسلاید بعدی نشان داده شده است.
- circleArray به کل آرایه ارجاع دارد.
- circleArray[1] به یک شیئ از Circle ارجاع دارد.

آرایه ای از اشیا

```
Circle[] circleArray = new Circle[10];
```



Summarizing the areas of the circles

آرایه ای از اشیا

```
public class TotalArea {
  /** Main method */
  public static void main(String[] args) {
    // Declare circleArray
    Circle3[] circleArray;
    // Create circleArray
   circleArray = createCircleArray();
    // Print circleArray and total areas of the circles
    printCircleArray(circleArray);
  }
  /** Create an array of Circle objects */
  public static Circle3[] createCircleArray() {
    Circle3[] circleArray = new Circle3[5];

    for (int i = 0; i < circleArray.length; i++) {
      circleArray[i] = new Circle3(Math.random() * 100);
    }
    // Return Circle array
    return circleArray;
  }
  /** Print an array of circles and their total area */
  public static void printCircleArray(Circle3[] circleArray) {
    System.out.printf("%-30s%-15s\n", "Radius", "Area");
    for (int i = 0; i < circleArray.length; i++) {
      System.out.printf("%-30f%-15f\n", circleArray[i].getRadius(),
        circleArray[i].getArea());
    }
    System.out.println("----------------------------------------");

    // Compute and display the result
    System.out.printf("%-30s%-15f\n", "The total areas of circles is",
      sum(circleArray));
  }
  /** Add circle areas */
  public static double sum(Circle3[] circleArray) {
    // Initialize sum
    double sum = 0;

    // Add areas to sum
    for (int i = 0; i < circleArray.length; i++)
      sum += circleArray[i].getArea();
    return sum;
  }
}
```