BKACAD®

HỌC VIỆN CÔNG NGHỆ BKACAD

Lập trình Java – Object & Class

Ths. Vũ Duy Khương

Nội dung

1. Khái niệm lập trình hướng đối tượng (OOP)
2. Tạo đối tượng & các biến tham chiếu đối tượng
3. Constructors
4. Phương thức, biến Class và Instance
5. Tầm tác dụng của biến
6. Từ khóa this
7. Case Studies

© Copyright 2020 FPT Software

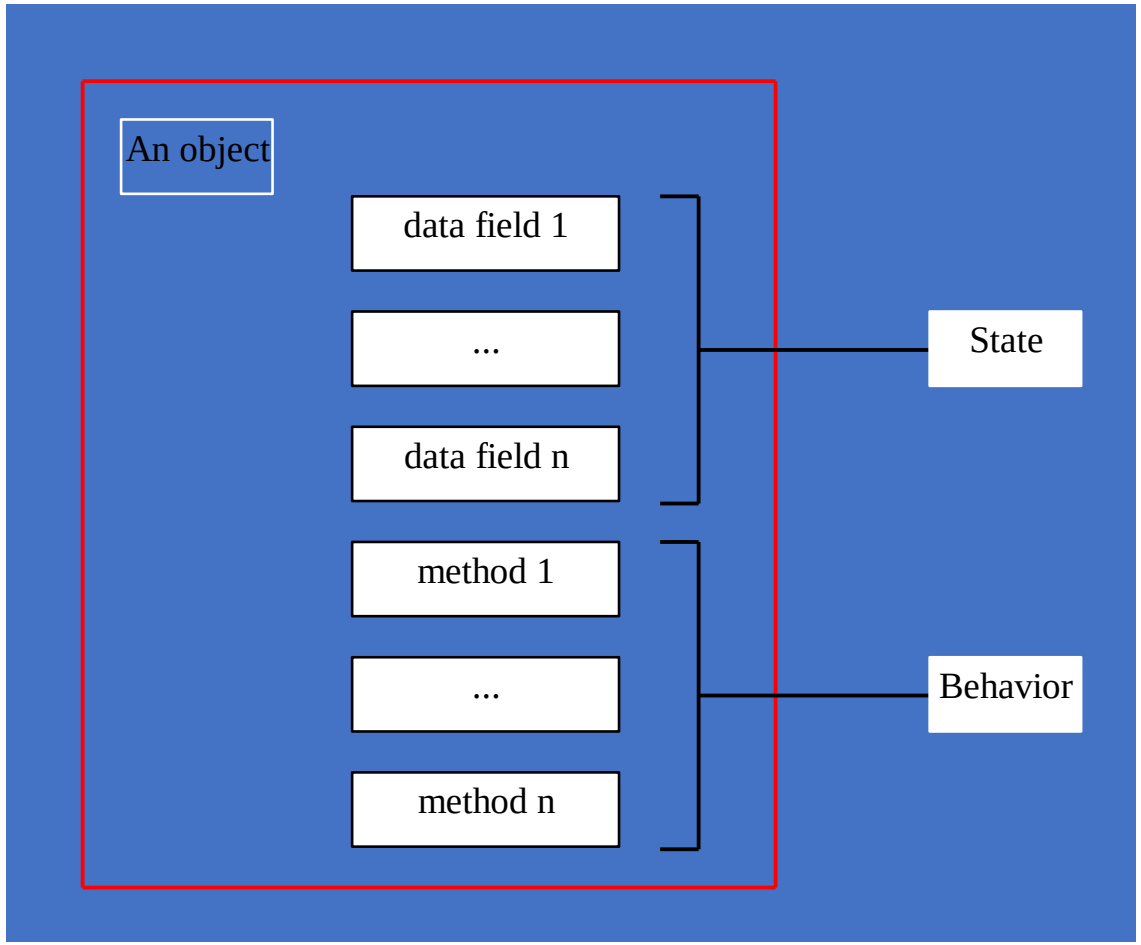
Đối tượng (Objects) và Lớp (Class)

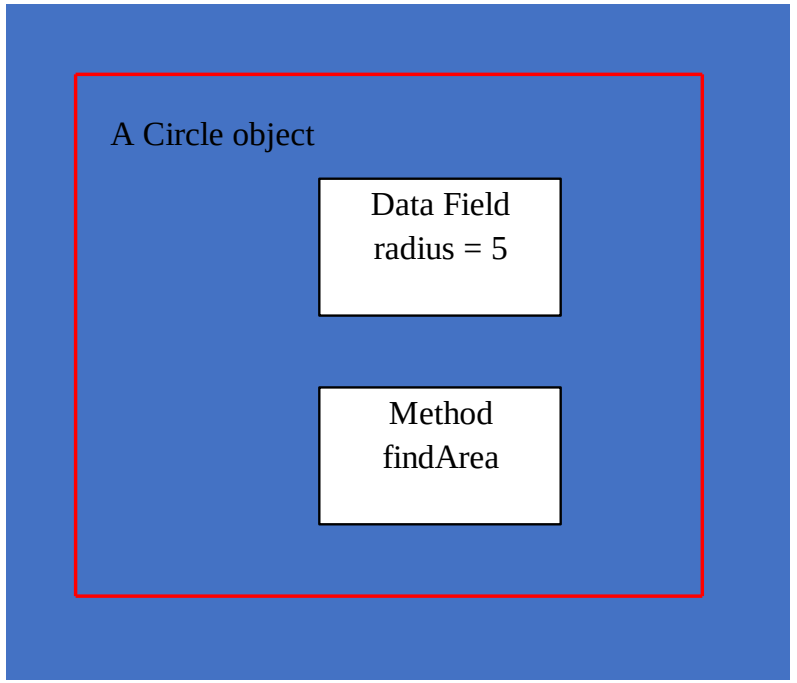
- Đối tượng đại diện cho một thực thể trong thế giới thật
- Ví dụ: 1 ô tô, 1 con người, 1 hình tròn, 1 khoản tiền
- Mỗi đối tượng có một `identity, state`, và các `behavior` duy nhất:
- `State` = tập các `data field` (`properties`)
- `Behavior` = tập các `method`
- `State` xác định đối tượng, `behavior` xác định đối tượng làm cái gì.

© Copyright 2020 FPT Software

Khái niệm lập trình hướng đối tượng

OOP - Lập trình sử dụng các đối tượng





Một đối tượng tổng quát

© Copyright 2020 FPT Software

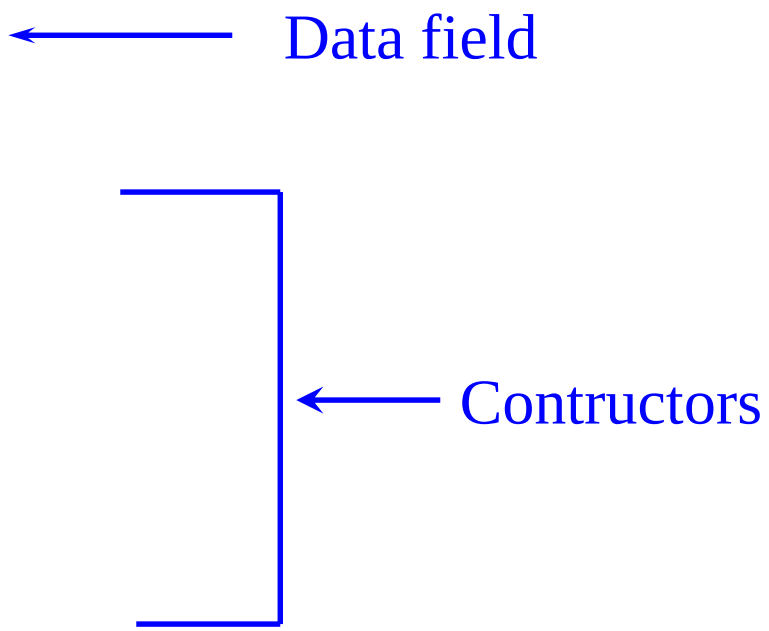
Khai báo lớp

```
class Circle {
  double radius = 1.0;

  Circle(){
  }

  Circle(double newRadius){
     radius = newRadius;
  }

  double findArea(){
     return radius * radius * 3.14159;
   }
}
```



© Copyright 2020 FPT Software

Khai báo biến tham chiếu đối tượng

```
ClassName objectReference;
```

Ví dụ:

```
Circle myCircle;
```

`myCircle` là 1 instance của lớp `Circle`.

© Copyright 2020 FPT Software

Tạo đối tượng

```
objectReference = new ClassName();
```

Ví dụ:

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

Tham chiếu đối tượng sẽ được gán cho biến `myCircle`.

© Copyright 2020 FPT Software

Khai báo/Tạo đối tượng trong 1 lệnh

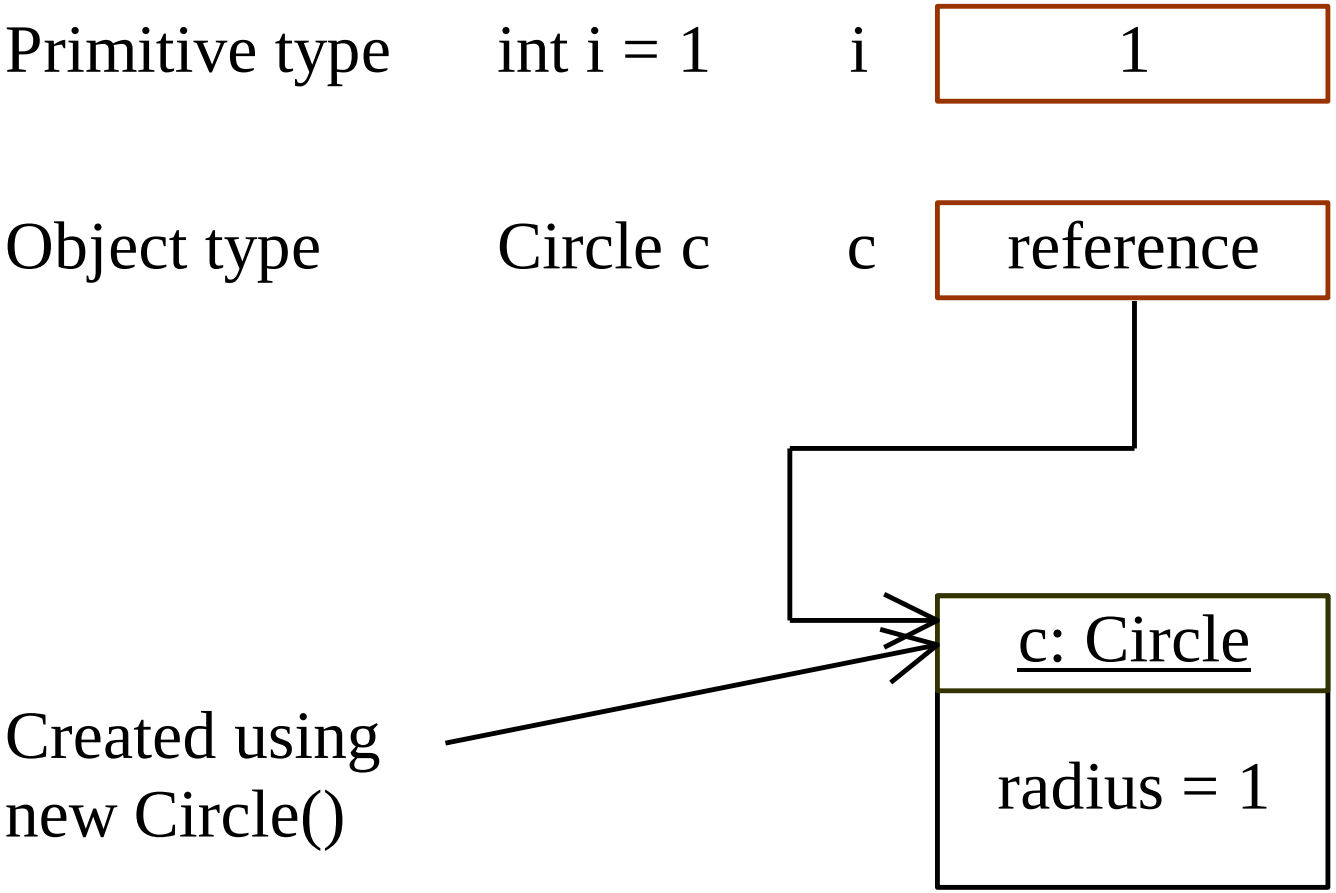
```
ClassName objectReference = new ClassName();
```

Ví dụ:

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

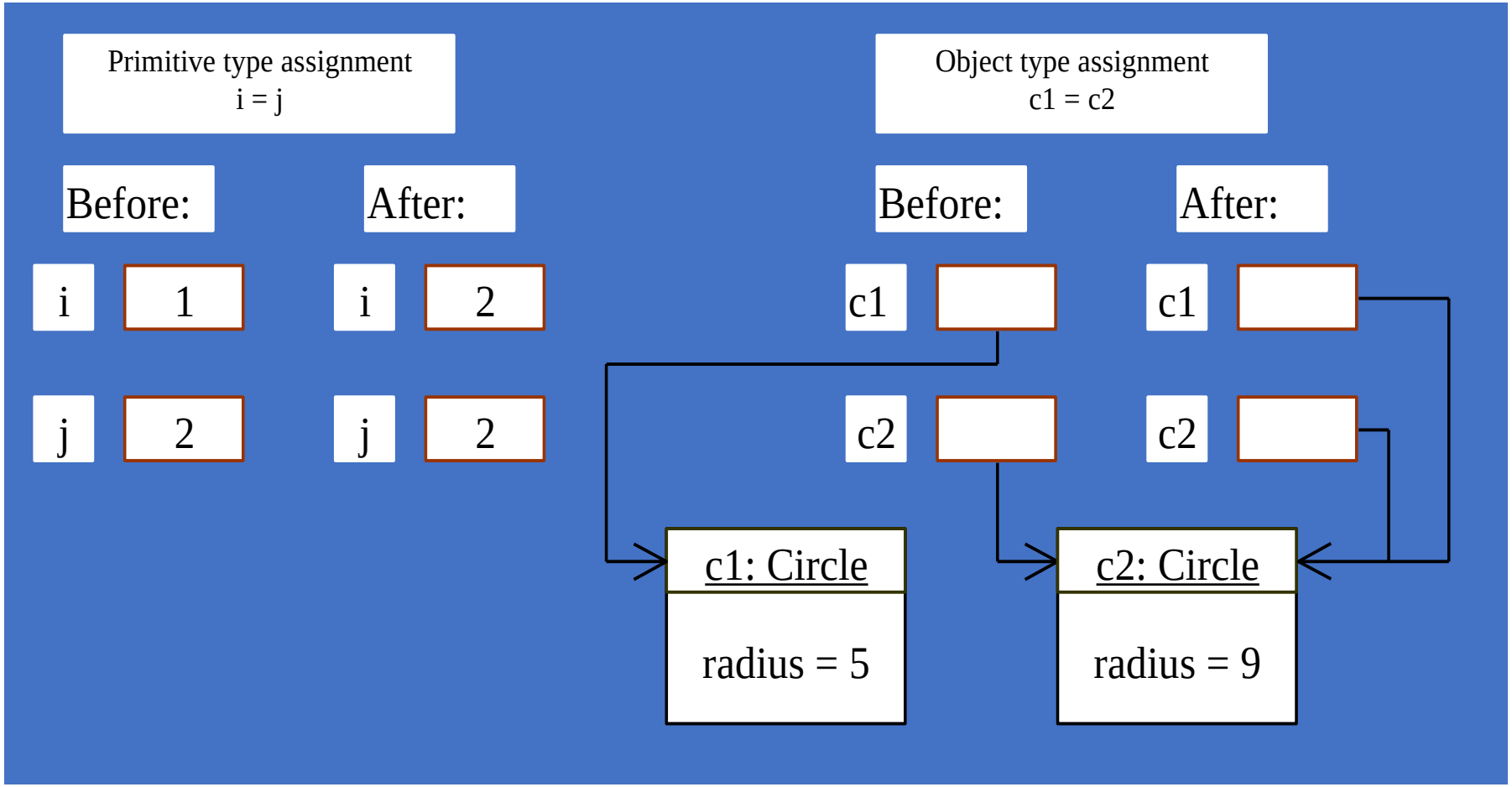
© Copyright 2020 FPT Software

Sự khác nhau giữa biến kiểu dữ liệu cơ sở & biến kiểu đối tượng



© Copyright 2020 FPT Software

Copy biến kiểu dữ liệu cơ sở và biến kiểu đối tượng



© Copyright 2020 FPT Software

Truy nhập đối tượng

- Tham chiếu dữ liệu của đối tượng:

```
objectReference.data
```

Ví dụ: `myCircle.radius`

- Gọi phương thức của đối tượng:

```
objectReference.method
```

Ví dụ: `myCircle.findArea()`

© Copyright 2020 FPT Software

Constructor

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

Circle() {
  radius = 1.0;
}

myCircle = new Circle(5.0);
```

Constructor là một dạng đặc biệt của phương thức, được gọi để xây dựng đối tượng.

© Copyright 2020 FPT Software

Constructor (tiếp)

- Một constructor không có tham số được gọi là *default constructor*.
- Các Constructor phải có cùng tên với class của nó.
- Các Constructor không có kiểu dữ liệu trả về, kể cả kiểu void.
- Các Constructor được gọi sử dụng toán tử `new` khi tạo một đối tượng.

Nó đóng vai trò khởi tạo đối tượng.

© Copyright 2020 FPT Software

Ví dụ 1: Sử dụng đối tượng

- Mục tiêu: Minh họa việc tạo đối tượng, truy nhập dữ liệu, sử dụng phương thức.

© Copyright 2020 FPT Software

Ví dụ 2: Sử dụng các lớp từ thư viện Java

- Mục tiêu: Minh họa việc sử dụng các lớp từ thư viện Java. Sử dụng lớp JFrame trong gói javax.swing để tạo 2 frame; sử dụng các phương thức trong lớp JFrame để thiết lập tiêu đề, kích thước, vị trí của các frame và hiển thị chúng.

© Copyright 2020 FPT Software

Ví dụ 2: Sử dụng các lớp từ thư viện Java

```
1  package jframedemo;
2  import javax.swing.JButton;
3  import javax.swing.JFrame;
4
5  public class JFrameDemo {
6      public static void main(String[] args) {
7          // TODO code application logic here
8          JFrame f = new JFrame();              // Khai bào và khởi tạo một Jframe
9
10         JButton b = new JButton("click");     // Khai bào và khởi tạo một JButton
11         b.setBounds(150, 50, 150, 150);
12
13         f.setTitle("Ví dụ Java Swing");
14         f.add(b);                             // thêm button vào JFrame
15         f.setSize(500, 300);                  // thiết lập kích thước cho của sổ
16         f.setLayout(null);                    // không sử dụng trình quản lý bố cục
17         f.setVisible(true);                   // hiển thị cửa sổ
18     }
19 }
```

© Copyright 2020 FPT Software

Ví dụ 3: Sử dụng Constructor

- Mục tiêu: Minh họa vai trò của các constructor và sử dụng chúng để tạo các đối tượng.
- Ví dụ:

```
class Bike {
    Bike() {
        System.out.println("Bike is created");
    }
    public static void main(String args[]) {
       Bike b=new Bike();
    }
}
```

© Copyright 2020 FPT Software

Các phương pháp truy nhập

Mặc định: các lớp, biến hoặc dữ liệu có thể được truy nhập bởi bất kỳ lớp nào trong cùng gói (package).

- **`public`**

  Lớp, dữ liệu, phương thức có thể được truy nhập bởi tất cả các lớp trong bất kỳ gói nào.

- **`private`**

  Dữ liệu hoặc phương thức chỉ có thể được truy nhập bởi lớp khai báo.

Các phương thức **`get`** và **`set`** được sử dụng để đọc và thay đổi các thuộc tính private

© Copyright 2020 FPT Software

Ví dụ 4: Sử dụng private & các phương thức truy nhập

Trong ví dụ này, radius sử dụng dữ liệu private, các phương thức truy nhập `getRadius` và `setRadius` được cung cấp để lấy và thay đổi radius.

© Copyright 2020 FPT Software

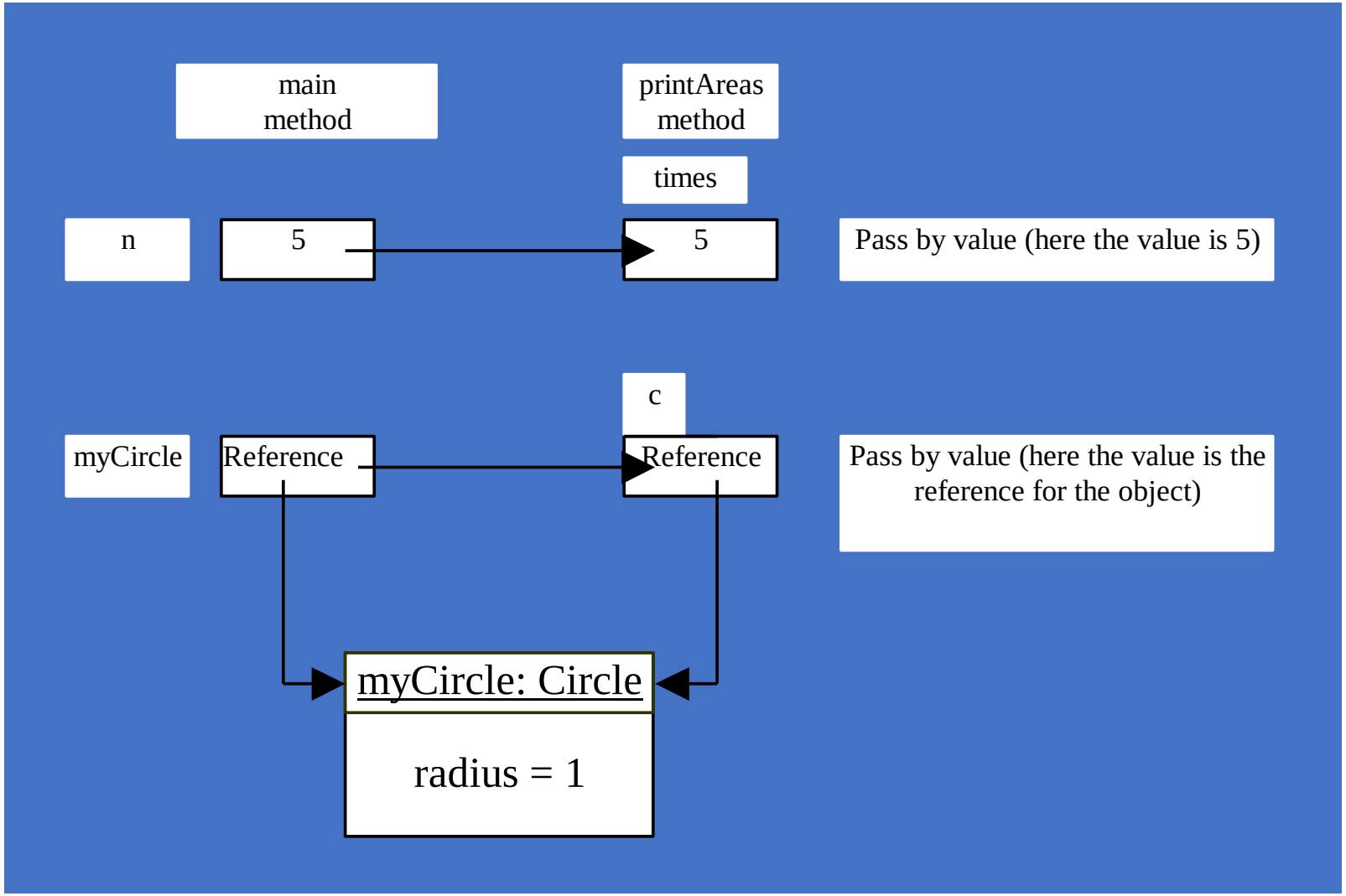
Truyền đối tượng cho phương thức

- Truyền tham trị (giá trị là tham chiếu tới đối tượng)

Ví dụ 5: Truyền tham số là đối tượng

© Copyright 2020 FPT Software

Truyền đối tượng cho phương thức (tiếp)



© Copyright 2020 FPT Software

Biến Instance

➢ Biến instance (vd: biến radius) thuộc một instance xác định, nó không được chia sẻ giữa các đối tượng trong cùng 1 class.

```
Circle circle1 = new Circle();
Circle circle2 = new Circle(5);
```

radius của circle1 độc lập với radius của circle2, được chứa ở những vùng nhớ khác nhau. Sự thay đổi radius của circle1 không ảnh hưởng tới radius của circle2

© Copyright 2020 FPT Software

Các biến, hằng, phương thức static

- Biến static được sử dụng để tạo biến class và các phương thức class mà có thể truy cập được mà không có instance của 1 class.

© Copyright 2020 FPT Software

Các biến, hằng, phương thức static (tiếp)

```
class ST_Employee
{
int eid;
String name;
static String company_name ="StudyTonight";
public void show()
{
System.out.println(eid+" "+name+" "+company_name);
}
public static void main( String[] args )
{
ST_Employee se1 = new ST_Employee();

se1.eid = 104;
se1.name = "Abhijit";
se1.show();
```

© Copyright 2020 FPT Software

Các biến, hằng, phương thức static (tiếp)

```
ST_Employee se2 = new ST_Employee();
se2.eid = 108;
se2.name = "ankit";
se2.show();
}
}
```

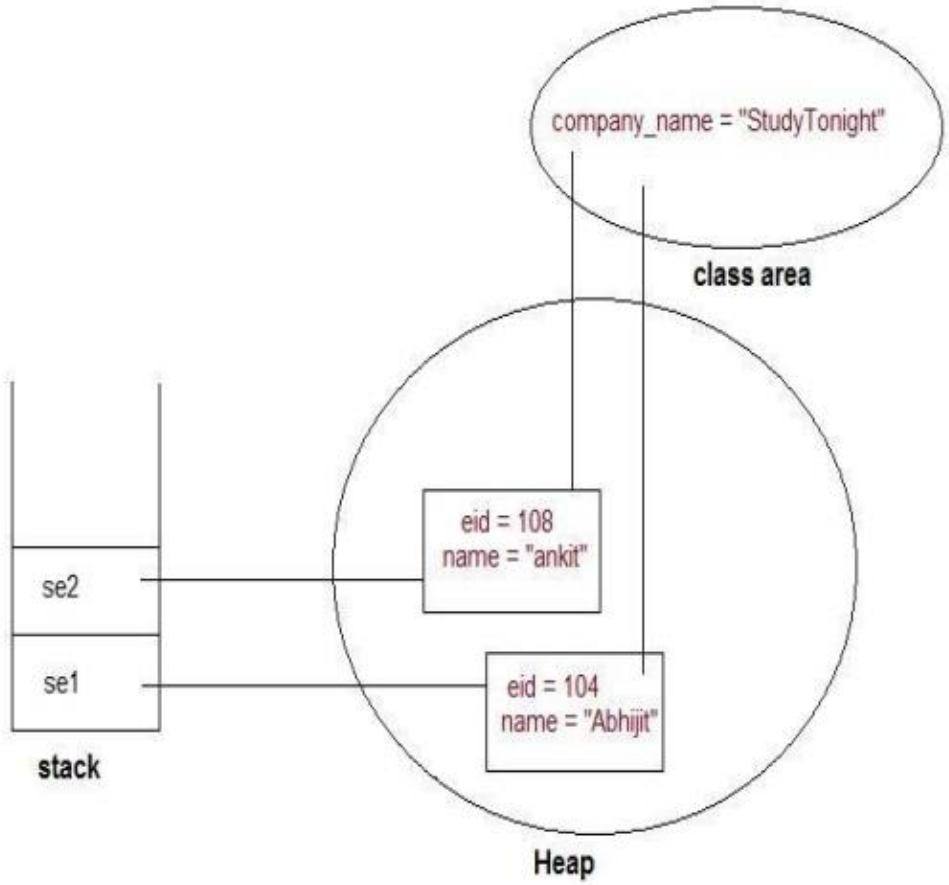
➔ **Output**

104 Abhijit StudyTonight

108 ankit StudyTonight

© Copyright 2020 FPT Software

Các biến, hằng, phương thức static (tiếp)



© Copyright 2020 FPT Software

Các biến, hằng, phương thức static (tiếp)

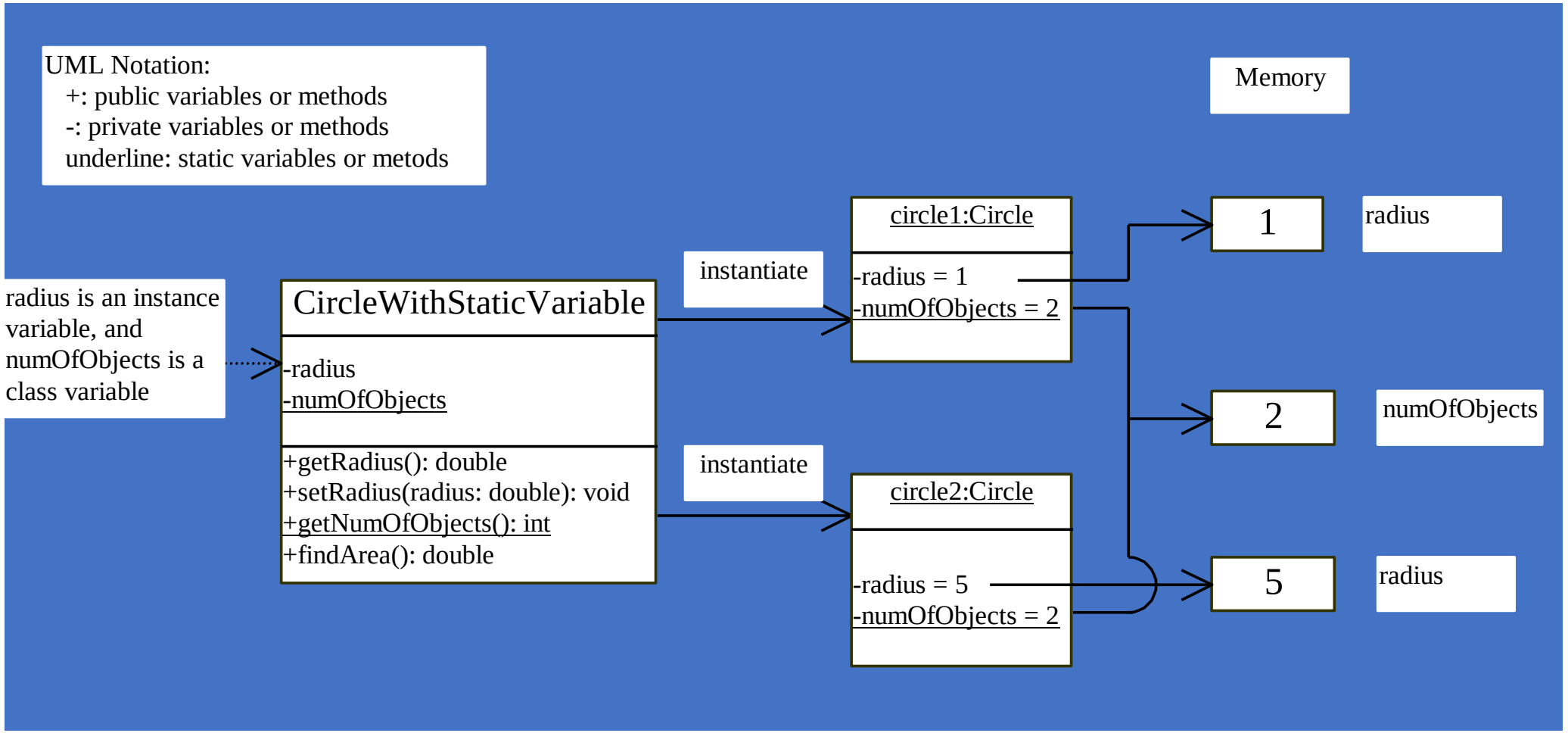
- Để các biến, hằng, phương thức trong class được chia sẻ cho tất cả các instance, sử dụng từ bổ nghĩa static trong khai báo.

- Biến static chứa giá trị trong vùng nhớ chung.

```
private static int numOfObj;
public static int getNumOfObj() {
     return numOfObj;
}
public final static double PI=3.1416;
```

© Copyright 2020 FPT Software

Các biến, hằng, phương thức static (tiếp)



© Copyright 2020 FPT Software

Ví dụ 5: Sử dụng biến, phương thức Instance và Static

Mục tiêu: Minh họa vai trò, cách sử dụng của các biến instance và static.

Biến static numOfObjects để theo dõi số đối tượng Circle được tạo.

© Copyright 2020 FPT Software

Phạm vi các biến

- Phạm vi của các biến static và instance là toàn bộ lớp. Chúng có thể được khai báo bất kỳ nơi nào trong lớp.
- Phạm vi của biến cục bộ bắt đầu từ khi khai báo đến hết block chứa biến đó. Một biến cục bộ phải được khai báo trước khi sử dụng.

© Copyright 2020 FPT Software

Từ khóa this

- Dùng `this` để:
  - thay thế cho đối tượng hiện tại.
  - gọi các constructor khác của đối tượng.

```
class Foo {
  int i = 5;

  void setI(int i){
     this.i = i
   }
}
```

```
public class Circle{
  private double rd;

  public Circle(double r){
       this.rd = r;
    }
}
```

© Copyright 2020 FPT Software

Mảng đối tượng

❖Khai báo và tạo mảng các đối tượng:

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

❖Khởi tạo:

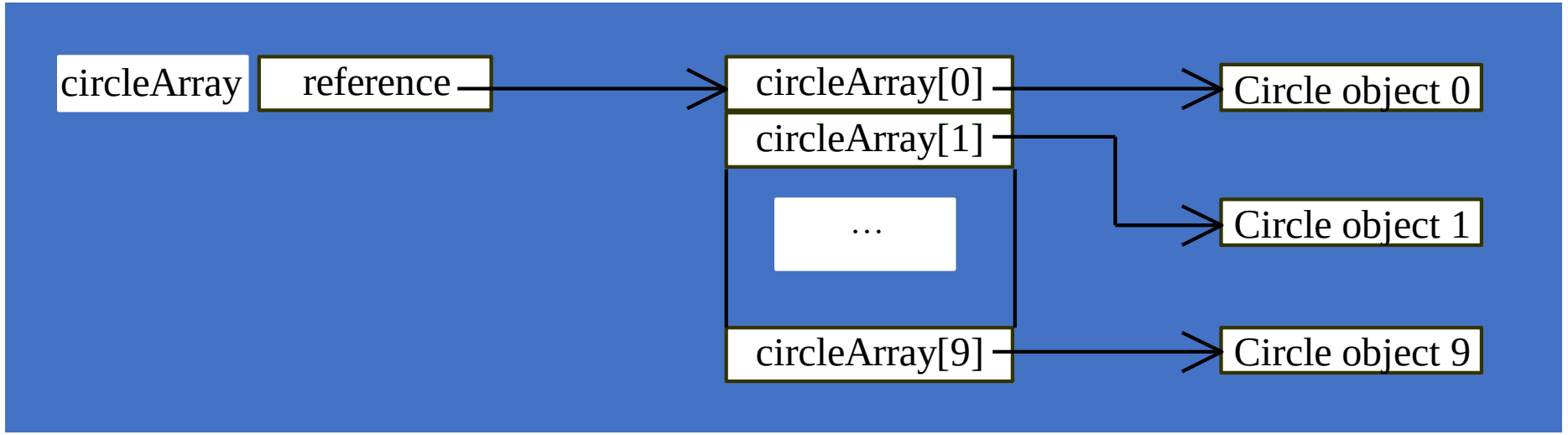
```
for(int i=0; i<circleArray.length; i++){
 circleArray[i] = new Circle();
}
```

© Copyright 2020 FPT Software

Mảng đối tượng (tiếp)

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

Một mảng các đối tượng thực chất là *mảng các biến tham chiếu*. Do đó gọi circleArray[1].findArea() sẽ gọi 2 mức tham chiếu: circleArray tham chiếu toàn bộ mảng, circleArray[1] tham chiếu tới một đối tượng Circle.



© Copyright 2020 FPT Software

Mảng đối tượng (tiếp)

Ví dụ 6: Tính tổng diện tích của các hình tròn

© Copyright 2020 FPT Software

Sự trừu tượng của lớp - Class Abstraction

- ✓ Sự thực hiện của lớp tách riêng với sự sử dụng nó.
- ✓ Người tạo lớp cung cấp sự miêu tả lớp để người sử dụng biết cách sử dụng.
- ✓ Người sử dụng không cần biết lớp được thực hiện như thế nào.

© Copyright 2020 FPT Software

Các lớp Java API và Core Java

- **`java.lang`**

  Chứa các lớp Java lõi (core Java class), gồm lớp số (numeric class), chuỗi ký tự, đối tượng. Gói này được import hoàn toàn vào tất cả các CT Java.

- **`java.awt`**

  Chứa các lớp đồ họa.

- **`java.applet`**

  Chứa các lớp hỗ trợ applet.

© Copyright 2020 FPT Software

Các lớp Java API và Core Java (tiếp)

- **`java.io`**

  Chứa các lớp cho các luồng vào-ra và các file.

- **`java.util`**

  Chứa nhiều tiện ích, ví dụ date.

- **`java.net`**

  Chứa các lớp hỗ trợ giao tiếp mạng.

© Copyright 2020 FPT Software

Các lớp Java API và Core Java (tiếp)

- **`java.awt.image`**
  Chứa các lớp giúp quản lý các ảnh bitmap.
- **`java.awt.peer`**
  Thực hiện Platform-specific GUI.
- Các lớp khác:

  `java.sql`
  `java.rmi`

© Copyright 2020 FPT Software

Q&A

© Copyright 2020 FPT Software

BKACAD®

HỌC VIỆN CÔNG NGHỆ BKACAD

THANK YOU