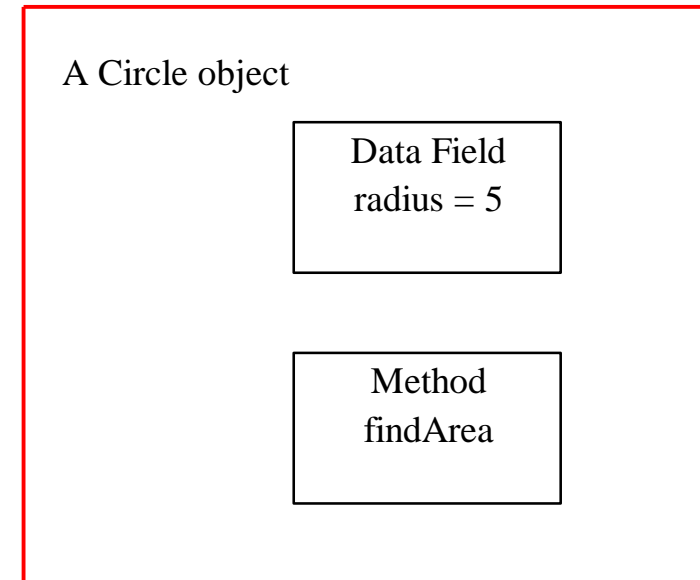
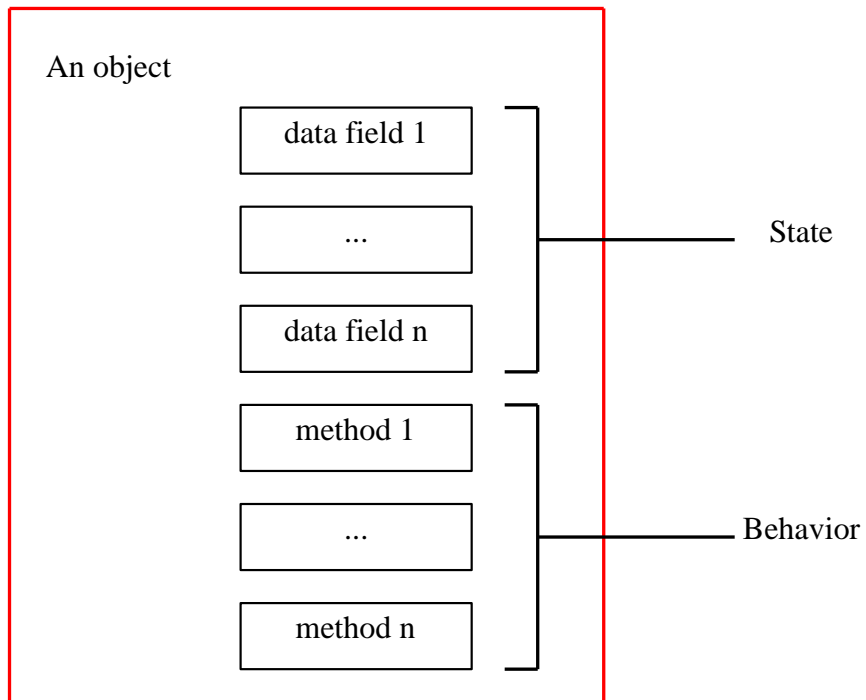




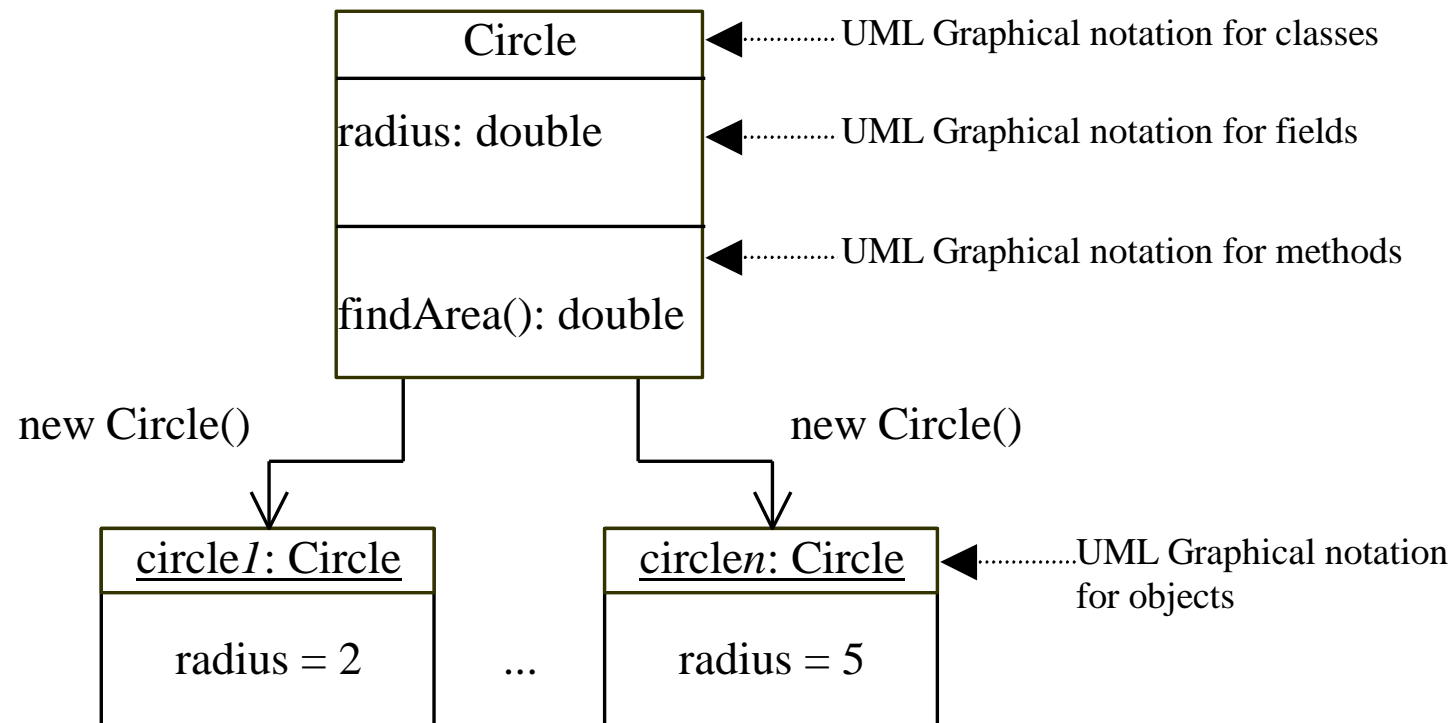
Lecture 2

FUNDAMENTALS OF OBJECT-ORIENTED PROGRAMMING

OO Programming Concepts



Class and Objects



Class declaration

```
class Circle {  
    double radius = 1.0;  
  
    double findArea(){  
        return radius * radius * 3.14159;  
    }  
}
```

Declaring Object Reference Variables

```
ClassName objectReference;
```

Example:

```
Circle myCircle;
```

Creating objects

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

Example:

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

The object reference is assigned to the object reference variable.

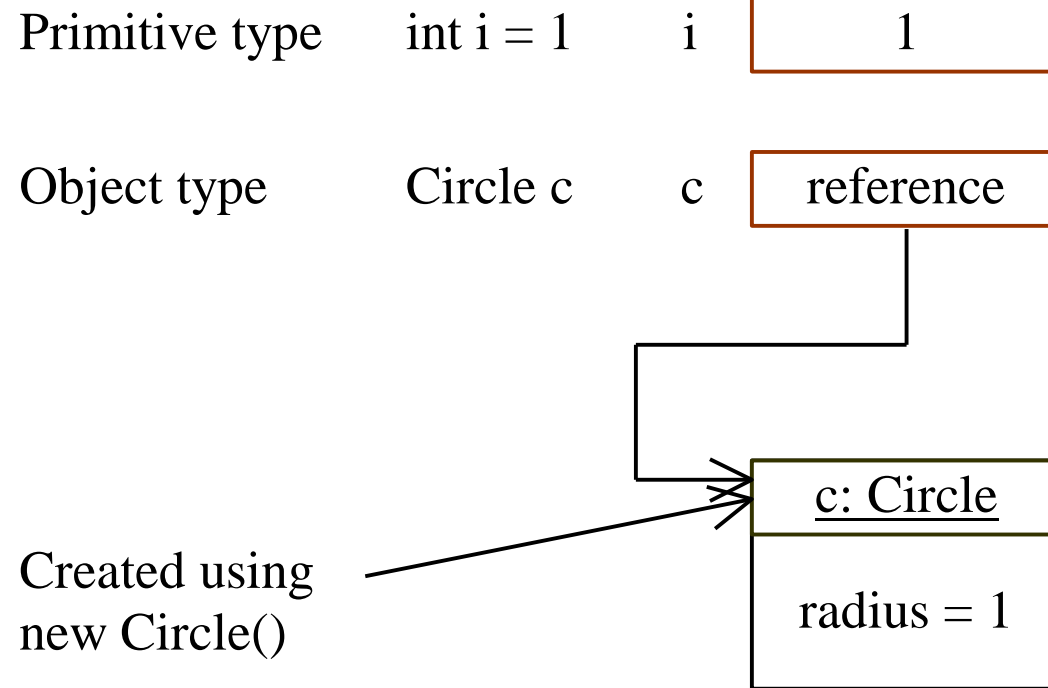
DECLARING / CREATING OBJECTS

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

Example:

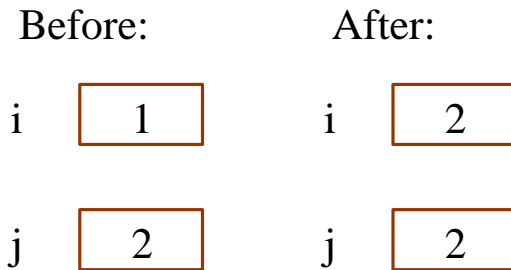
```
Circle myCircle = new Circle();
```

Differences between variables of primitive Data types and object types

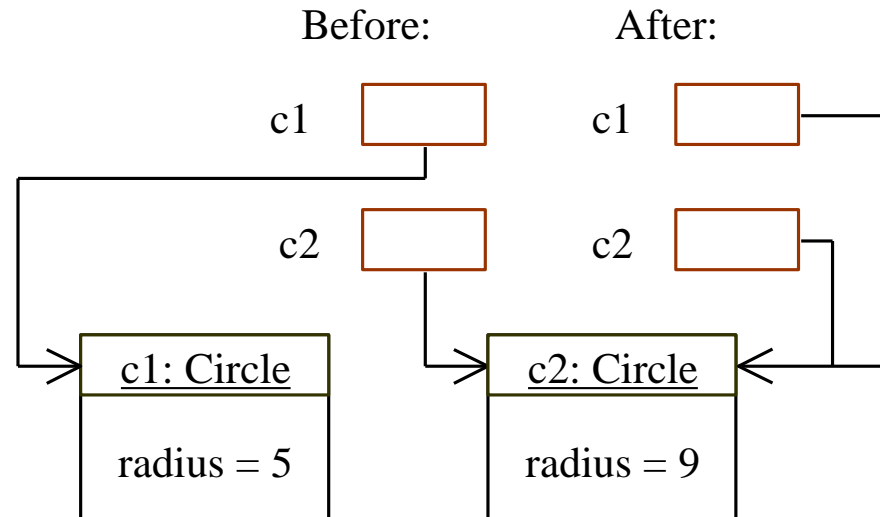


Copying Variables of Primitive Data Types and Object Types

Primitive type assignment
 $i = j$



Object type assignment
 $c1 = c2$



Modifiers Static vs. Instance Members

- ▶ By default, members are per instance
 - ▶ Each instance gets its own fields
 - ▶ Methods apply to a specific instance
- ▶ Static members are per type
 - ▶ Static methods can't access instance data
 - ▶ No `this` variable in static methods

Static methods

- ▶ Static methods are called by using the class name, not the instance of the class.
- ▶ The Console class and its Read and Write methods are an example of static methods. The following code example calls Console.WriteLine and Console.ReadKey methods without creating an instance of the Console class.

```
class Program
{
    public static void withoutObj()
    {
        Console.WriteLine("Hello");
    }
    static void Main()
    {
        Program.withoutObj();
        Console.ReadKey();
    }
}
```

Access modifiers

- ▶ Access modifiers specify who can use a type or a member
- ▶ Access modifiers control encapsulation
- ▶ Class members can be public, private, protected, internal, or protected internal
- ▶ Struct members can be public, private or internal

Access modifiers

If the access modifier is	Then a member defined in type T and assembly A is accessible
public	to everyone
private	within T only
protected	to T or types derived from T
internal	to types within A
protected internal	to T or types derived from T -or- to types within A

Access defaults

- ▶ You should always explicitly mark what access you want.
- ▶ Class definitions default to internal.
- ▶ Member fields, methods and events default to private for classes