SLIIT

Discover Your Future

IT2020
Object Oriented Programming

Lecture 03

Object Oriented Concepts – Part 2

Dr. Kalpani Manathunga

SLIIT
FACULTY OF COMPUTING

Learning Outcomes

In the previous lecture,

- Classes and Objects
- Abstraction
- Encapsulation

At the end of the Lecture you should know

- Get and set methods
- Inheritance
- Polymorphism

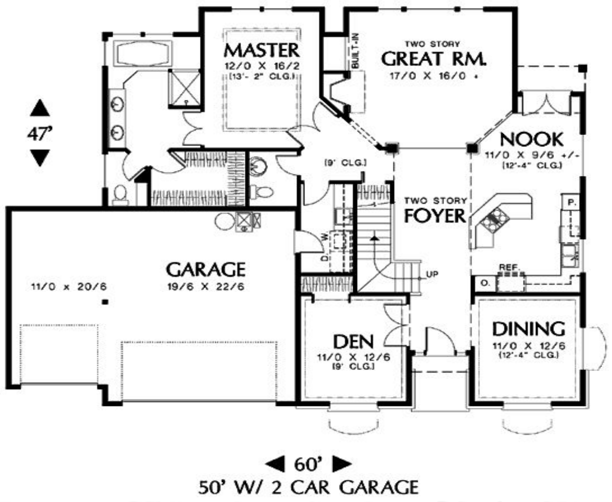
Object Oriented Programming

- Complex problems are broken into smaller sub systems or modules, each solving a particular sub problem

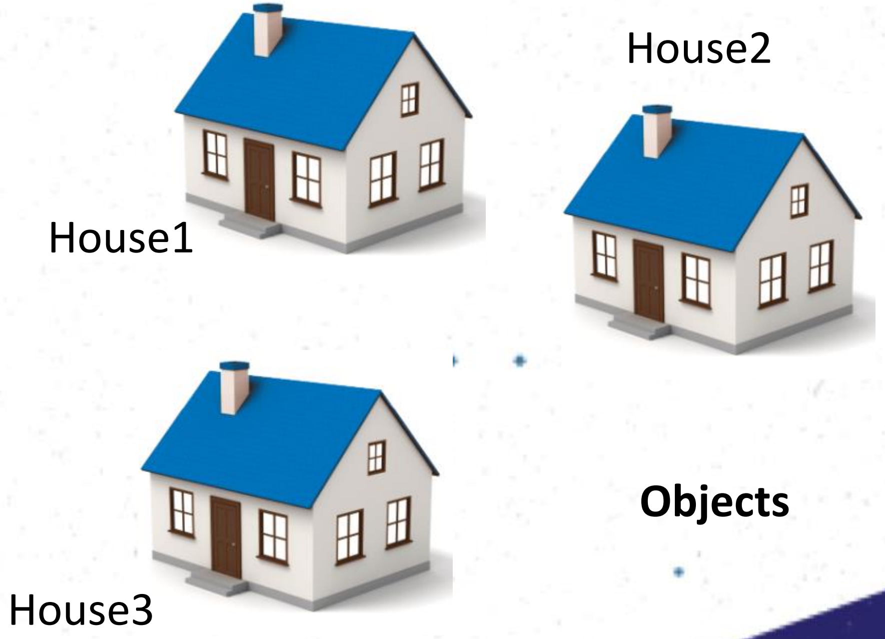
- Set of objects interact with one another
- Objects are derived from class definitions, contain data and methods

Classes and Objects

- A Class is an entity described using data members and methods
- An Object is a specific instance of the data type (class)

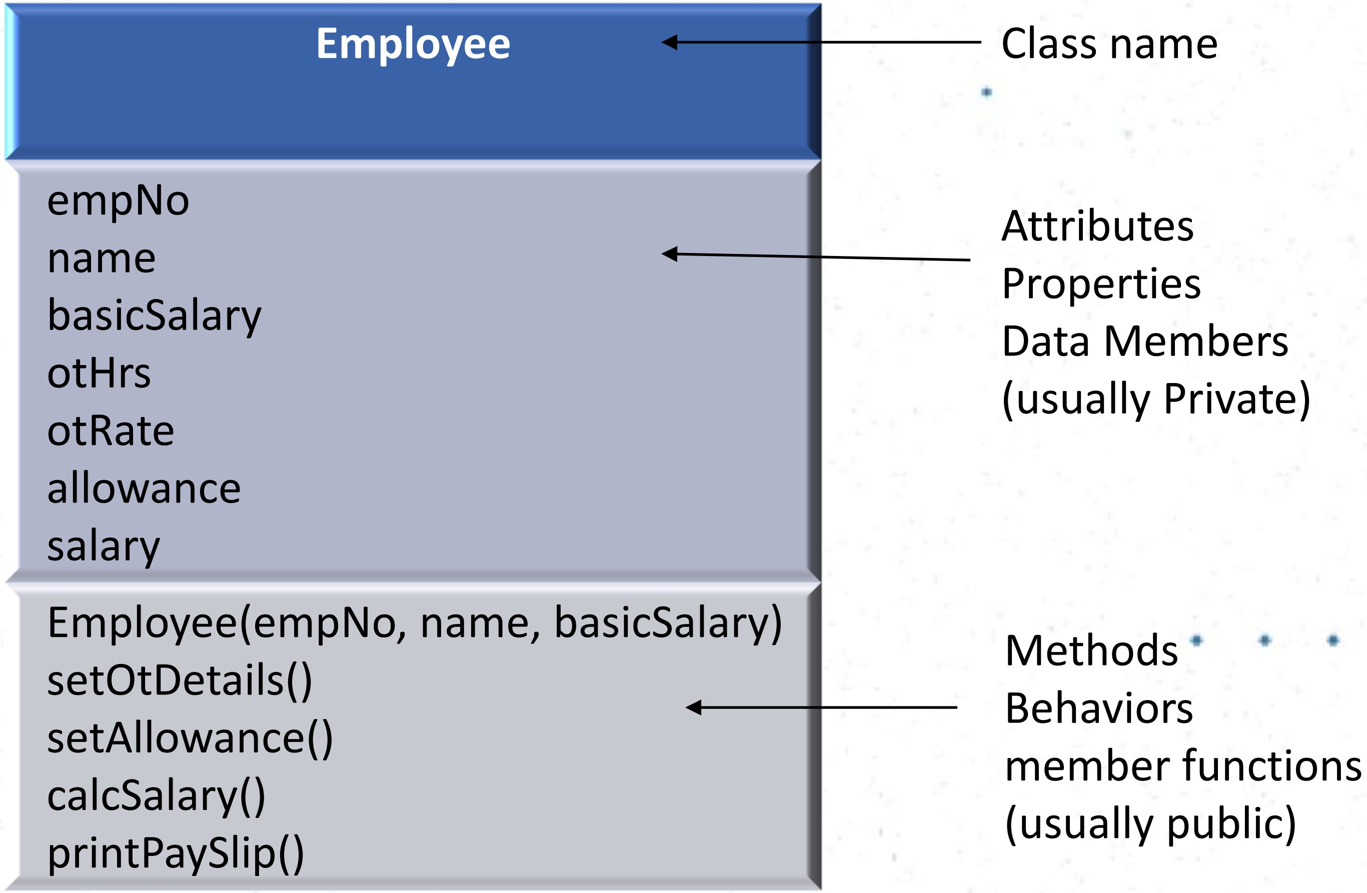


Class House



Objects

Terminology



Employee

empNo
name
basicSalary
otHrs
otRate
allowance
salary

Employee(empNo, name, basicSalary)
setOtDetails()
setAllowance()
calcSalary()
printPaySlip()

Class name

Attributes
Properties
Data Members
(usually Private)

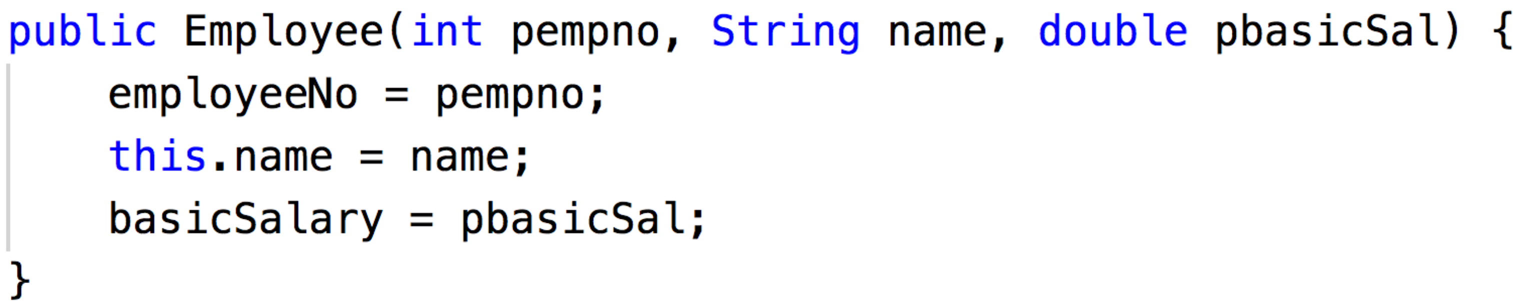
Methods
Behaviors
member functions
(usually public)

Getters and Setters

- In general properties are declared as *private* preventing them from being accessed from outside the class
- Typically an attribute will have a getter (accessor) - a get method to return its value
- And a setter (mutator) - a set method to set a value
  - e.g. a property called length will have a getter defined as int getLength() and a setter defined as void setLength(). Both these methods will be declared as public methods.

“this” keyword

- “this” **keyword** can be used to refer to any member of the current object within an instance method or a constructor.



```java
public Employee(int pempno, String name, double pbasicSal) {
    employeeNo = pempno;
    this.name = name;
    basicSalary = pbasicSal;
}
```

- We need to use this.name to refer to the property name to distinguish it from the parameter name.

Exercise

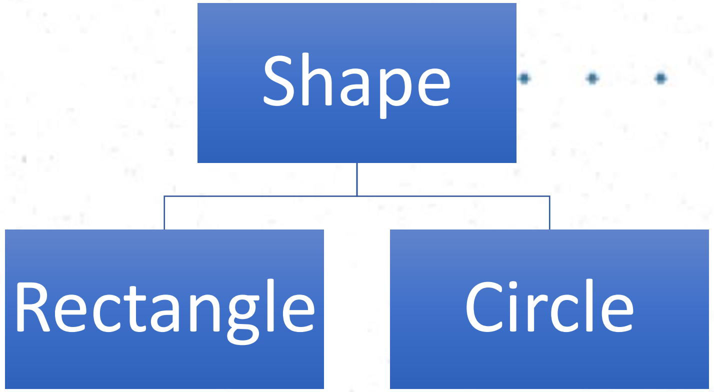
- Add getters and setters to the Receptionist class.

Exercise – Sample Answer

```
public class Receptionist{
    …………..
    public Receptionist(int pID, String pName, String pTelephoneNumber){
        this.staffID = pID
        this.name= pName ;
        this. telephoneNumber = pTelephoneNumber;
    }
    public void setID(int ID){
        this.staffID =ID;
    }
    public int getID(){
        return this.staffID;
    }
}
```

Generalization/Inheritance

- Inheritance is a mechanism in which one object acquires all properties and behaviors of a parent object
- Child class **is a** type of the parent class
- Inheritance promotes code reusability
- Child classes “inherit” the attributes and methods defined in the parent class


Shape

Rectangle

Circle

C++ vs Java - Inheritance

C++

```
class Circle : public Shape {
```

Java

```
class Circle extends Shape {
```

Java has a simpler inheritance mechanism where base class is extended as public.

C++ has multiple inheritance compared to Java's Single Inheritance.

C++ vs Java - Inheritance

C++

```
Circle (string tname, int r) : Shape ( tname) {
        radius = r;
}
```

Java

```
public Circle (String tname, int r) {
    super(tname);
    radius = r;
}
```

When you want to call a base class constructor C++ Requires to explicitly name the base class. In Java we use the super keyword to access the direct descendent class.

However this implies that in Java you can’t directly call a class higher in the hierarchy e.g. the Grandfather class which is not in C++

C++ vs Java - Inheritance

C++

```
virtual void speak() {}
```

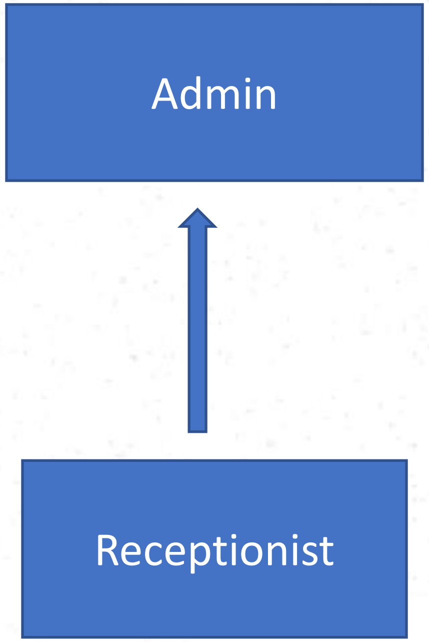
Java

```
public void speak() {}
```

- All methods in Java are virtual by default. Hence, methods are overridable
- In C++ we need to explicitly define polymorphic methods using virtual keyword. Then only sub classes can override such methods.

Example

Think of a way to implement generalization for our example scenario, Hospital Management System



Example from Receptionist class

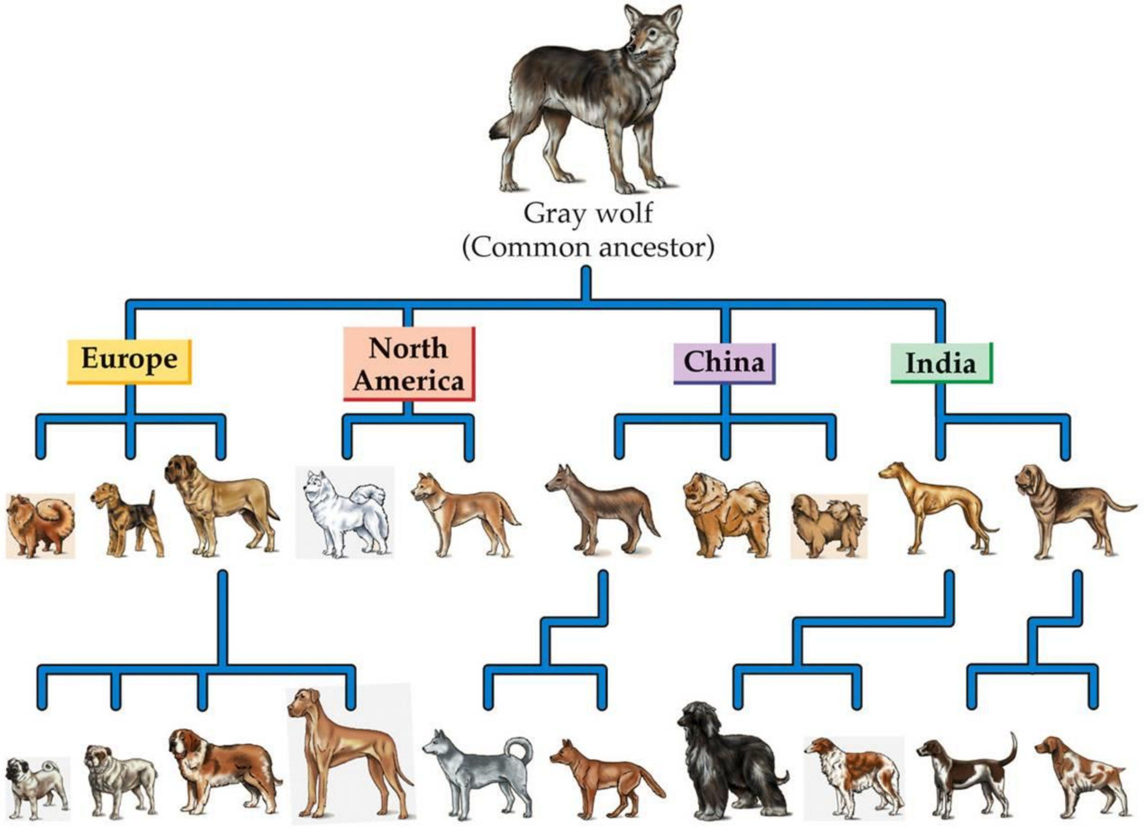
```
public class Admin{
          protected int staffID;
          protected String name;
          }

public class Receptionist extends Admin{
          private String telephoneNumber;


}
```

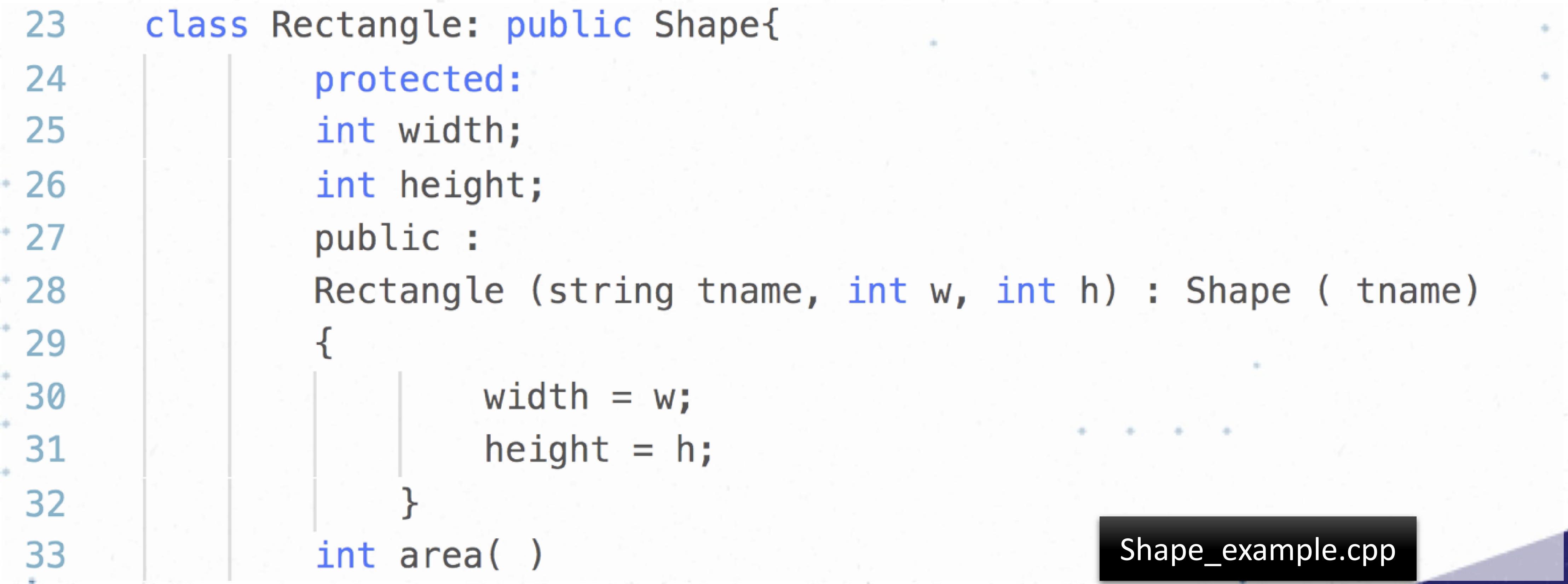
Object Class

- A class hierarchy is like the taxonomy of animals that shows their ancestry.
- There are many breeds of dogs. A well-known fact is that the common origin of a dog is the Gray Wolf.
- All Java classes are derived from a class called **Object**.
- This includes all the existing Java built in classes and the classes that you write.
- The methods and properties of the Object class are accessible to any Java class that you create.
- Object class in the Javadoc



DISCOVER BIOLOGY, Second Edition, Chapter 21 Box © 2002 Sinauer Associates, Inc., and W. W. Norton and Company

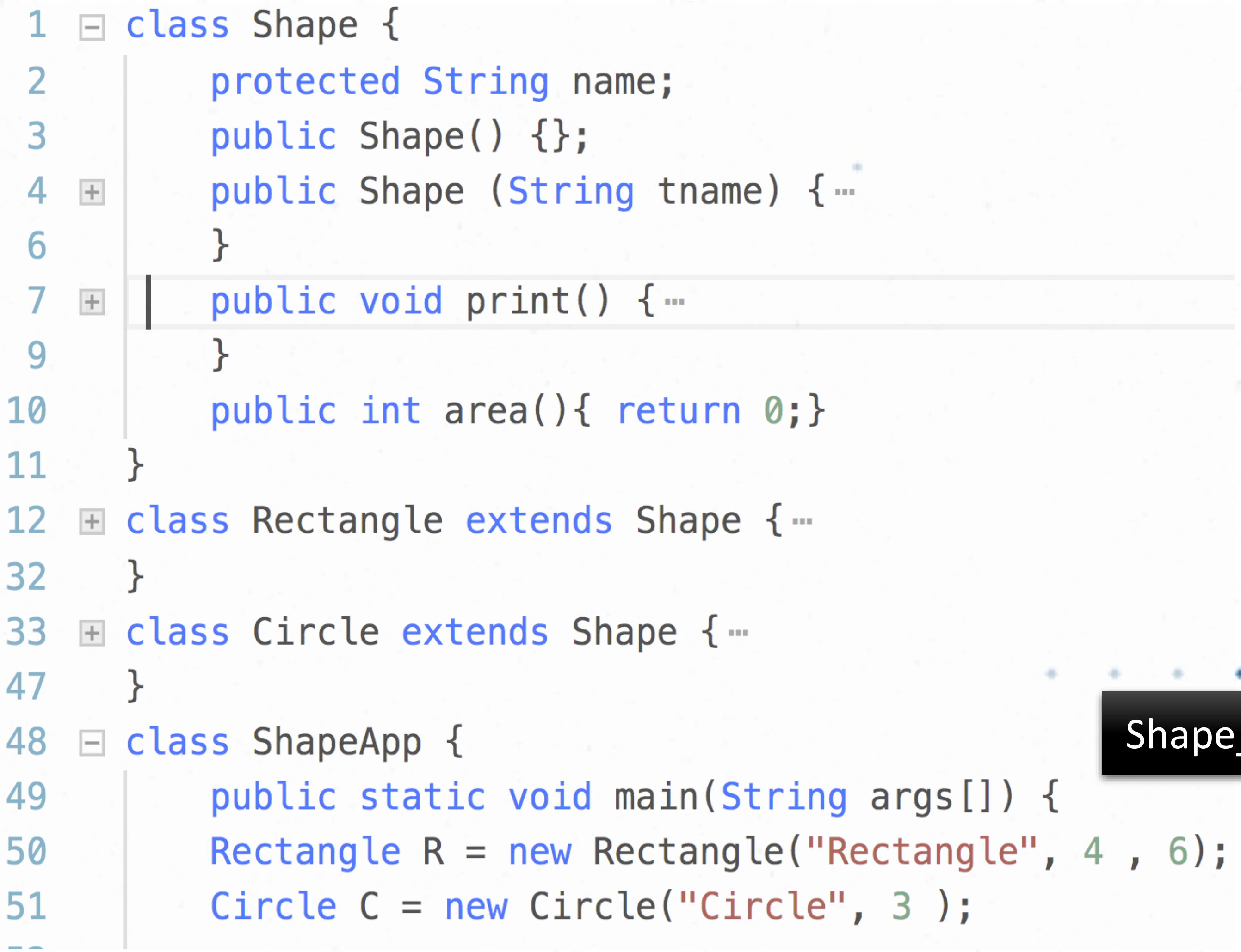
Inheritance Shapes Example C++



```cpp
class Rectangle: public Shape{
        protected:
        int width;
        int height;
        public :
        Rectangle (string tname, int w, int h) : Shape ( tname)
        {
                width = w;
                height = h;
            }
        int area( )
```

Shape_example.cpp

Inheritance Shapes Example Java



```java
class Shape {
    protected String name;
    public Shape() {};
    public Shape (String tname) {⋯
    }
    public void print() {⋯
    }
    public int area(){ return 0;}
}
class Rectangle extends Shape {⋯
}
class Circle extends Shape {⋯
}
class ShapeApp {
    public static void main(String args[]) {
    Rectangle R = new Rectangle("Rectangle", 4 , 6);
    Circle C = new Circle("Circle", 3 );
```

Shape_example.java

Polymorphism

- Ability to assign a different meaning or usage to something in different contexts
- Ability of an object to take many forms. Common with Inheritance concept
- Consider the request (analogues to a method) “*please cut this in half“* taking many forms

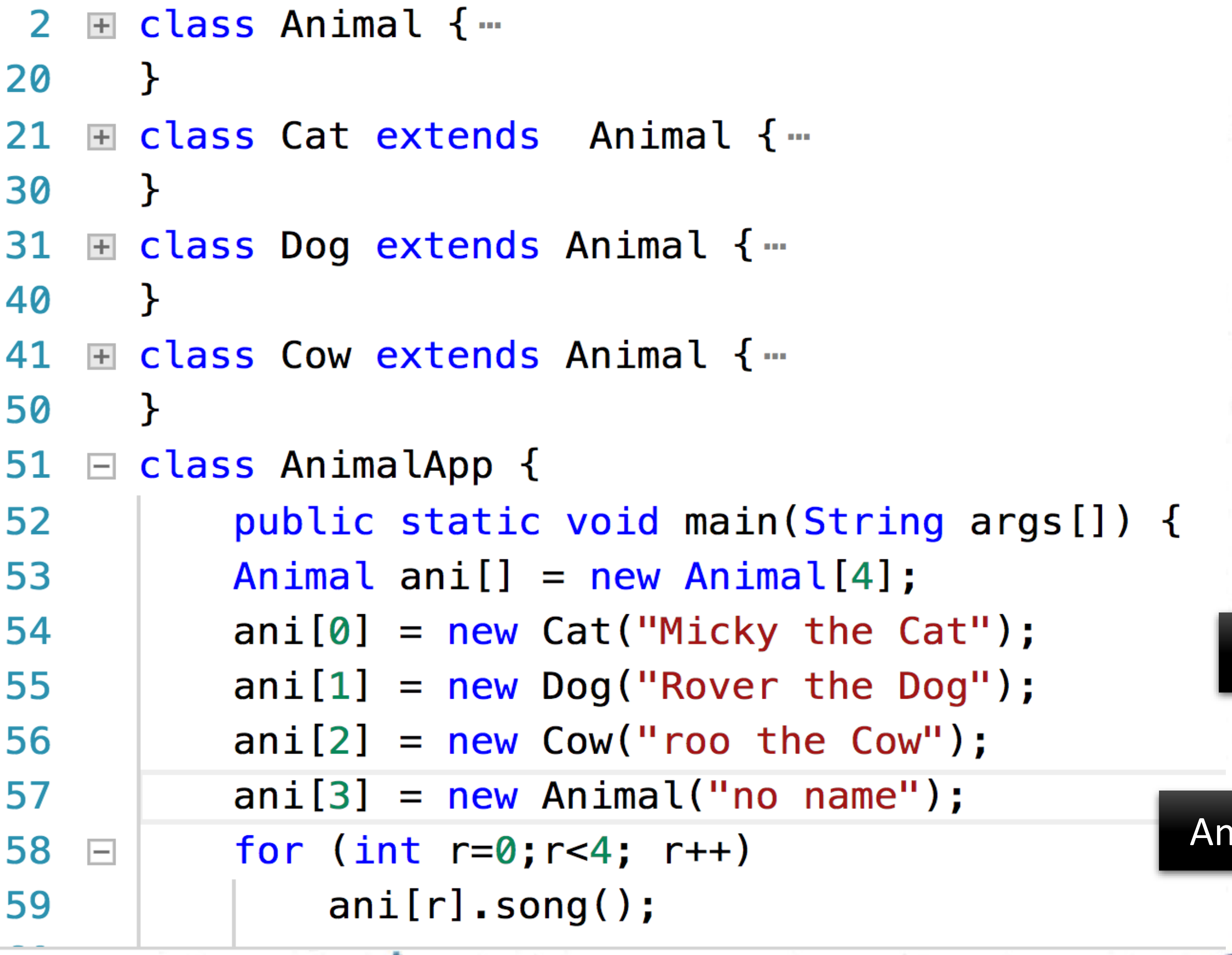
For a cake:

- Use a knife
- Apply gentle pressure

For a cloth:

- Use a pair of scissors
- Move fingers in a cutting motion

Animal Example



```java
class Animal {⋯
}
class Cat extends  Animal {⋯
}
class Dog extends Animal {⋯
}
class Cow extends Animal {⋯
}
class AnimalApp {
    public static void main(String args[]) {
    Animal ani[] = new Animal[4];
    ani[0] = new Cat("Micky the Cat");
    ani[1] = new Dog("Rover the Dog");
    ani[2] = new Cow("roo the Cow");
    ani[3] = new Animal("no name");
    for (int r=0;r<4; r++)
        ani[r].song();
```

Animal_example.cpp

Animal_example.java

Thank you!