Lecture 03 (Jav)

Classes & Objects

CPE112 Programming with Data Structures 2 / 2024

[29th Jan 2025]

Recall struct from C

```c
#include<stdio.h>
#include<string.h>

typedef struct ball{
    char color[15];
    char name[20];
    double radius;
}BALL_T;


int main(){
    BALL_T football = {"White", "Taweewie", 10.3};
    printf("%s", football.name);
    return 0;
}
```

A struct is a collection of variables (data) under one name

A ball should have an action

```
typedef struct ball{
    char color[15];
    char name[20];
    double radius;
}BALL_T;
```

```
void bounce(BALL_T ball){
    printf("Ball %s BOING!\n", ball.name);
}

void paintTo(BALL_T ball, char* newColor){
    strcpy(ball.color, newColor);
    printf("%s is now %s\n", ball.name, ball.color);
}

int main(){
    BALL_T football = {"White", "Taweewie", 10.3};
    bounce(football);
    paintTo(football, "Pink");
    return 0;
}
```

Implement a ball ADT using “Class”

- Fields:
  - Color
  - Name
  - Radius
- Operations:
  - Bounce up
  - Change the color

We group balls’ fields and operation under a “class”

Java code for class “Ball”

```java
public class Ball {
    private String color;
    private String name;
    private double radius;

    public Ball(String color, String name, double radius){
        this.color = color;
        this.name = name;
        this.radius = radius;
    }
    public void bounce(){
        System.out.println("Ball "+this.name+" BOING!");
    }
    public void paintTo(String newColor){
        this.color = newColor;
        System.out.println(this.name+" is now "+this.color);
    }
    public static void main(String[] args){
        Ball football = new Ball("White","Taweewie",10.3);
        football.bounce();
        football.paintTo("Pink");
    }
}
```

Fields / Attributes

Constructor

Methods

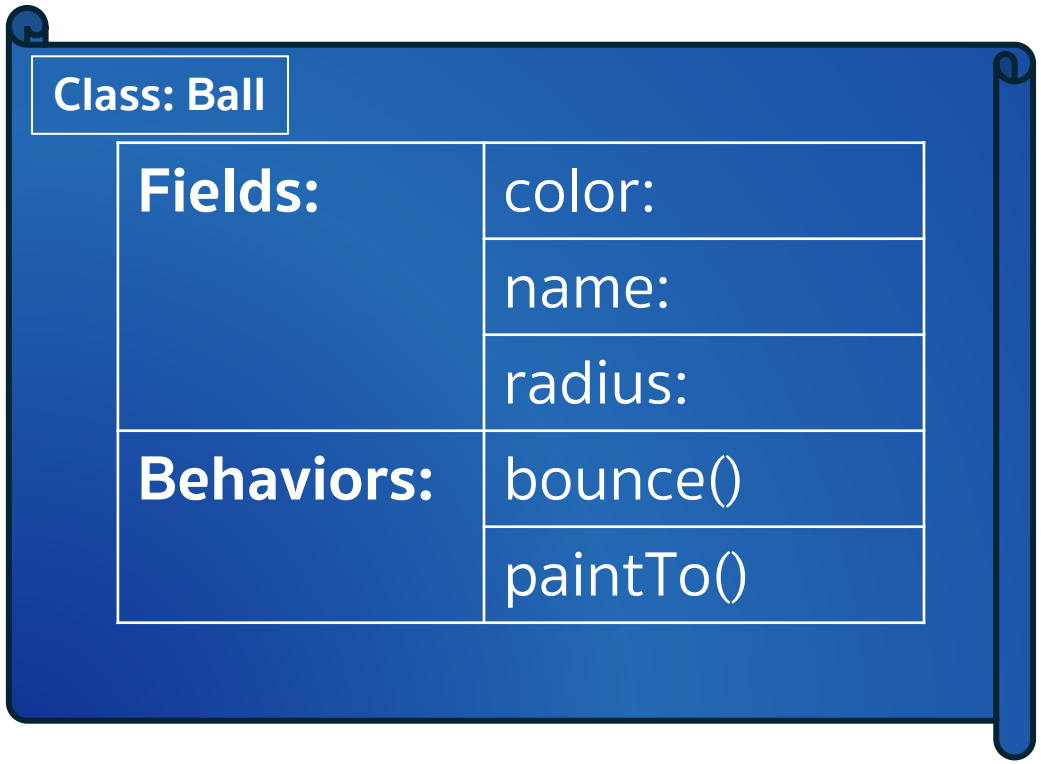
Java Classes

Java classes

- A class is a **<u>user-defined data type</u>** similar to a struct, but it can contain FUNCTIONS (unlike structs which **traditionally** cannot)
- A class is a blueprint to group of **objects** having similar properties. It defines the structures and behavior of objects.

Java classes

- You can see this is an empty blueprint here
- This blueprint here will be used to create **Objects**



Disclaimer: This is not Class diagram

Java Objects

Java objects

- An object is an instance of class that is created to use the attributes and method of a class.
- A class can create many objects.

Creating objects

```java
public class Ball {
    private String color;
    private String name;
    private double radius;

    public Ball(String color, String name, double radius){
        this.color = color;
        this.name = name;
        this.radius = radius;
    }
    public void bounce(){
        System.out.println("Ball "+this.name+" BOING!");
    }
    public void paintTo(String newColor){
        this.color = newColor;
        System.out.println(this.name+" is now "+this.color);
    }
    public static void main(String[] args){
        Ball football = new Ball("White","Taweewie",10.3);
        Ball basketball = new Ball("Orange", "Lovely", 45.7);
        Ball volleyball = new Ball("Yellow and blue", "Hinata", 21.3);
    }
}
```

Creating objects

```java
public class Ball {
    private String color;
    private String name;
    private double radius;

    //...
    //methods
    //...

    public static void main(String[] args){
        Ball football = new Ball("White","Taweewie",10.3);
        Ball basketball = new Ball("Orange", "Lovely", 45.7);
        Ball volleyball = new Ball("Yellow and blue", "Hinata", 21.3);
    }
}
```

Creating objects

Class: Ball

Fields:	color: “White”
	name: “Taweewie”
	radius: 10.3
Behaviors:	bounce()
	paintTo()

Object: football

Class: Ball

Fields:	color: “Orange”
	name: “Lovely”
	radius: 45.7
Behaviors:	bounce()
	paintTo()

Object: basketball

- Such a process is called **INSTANTIATION** [Instance = ตัวอย่าง]
- The object is allocated in the memory and set up its structures according to class definition

Java Constructors

Java constructor

```java
public class Ball {
    private String color;
    private String name;
    private double radius;

     public Ball(String color, String name, double radius){
         this.color = color;
         this.name = name;
         this.radius = radius;
     }
    public void bounce(){
        System.out.println("Ball "+this.name+" BOING!");
    }
    public void paintTo(String newColor){
        this.color = newColor;
        System.out.println(this.name+" is now "+this.color);
    }
    public static void main(String[] args){
         Ball football = new Ball("White","Taweewie",10.3);
        football.bounce();
        football.paintTo("Pink");
    }
}
```

Constructor

The new keyword

What are constructors?

- A constructor is a **special method** used to **INITIALIZE** object
- It is used together with new keywords.
- These are key features of constructures:
  - It must be the same name with the class.
  - No return type
  - It is called automatically when the object is created.

What are constructors?

```
public class Ball {
    private String color;
    private String name;
    private double radius;

    public Ball(String color, String name, double radius){
        this.color = color;
        this.name = name;
        this.radius = radius;
    }

    //...

    public static void main(String[] args){
        Ball football = new Ball("White","Taweewie",10.3);
        Ball basketball = new Ball("Orange", "Lovely", 45.7);
    }
}
```

The constructor and class name must be the same

No return type

Called automatically when object is created

What are constructors?

```
public class Ball {
    private String color;
    private String name;
    private double radius;

    public Ball(String color, String name, double radius){
        this.color = color;
        this.name = name;
        this.radius = radius;
    }

    //...

    public static void main(String[] args){
        Ball football = new Ball("White","Taweewie",10.3);
        Ball basketball = new Ball("Orange", "Lovely", 45.7);
    }
}
```

The this keyword

Java Modifiers

Java modifiers

```java
public class Ball {
    private String color;
    private String name;
    private double radius;

    public Ball(String color, String name, double radius){
        this.color = color;
        this.name = name;
        this.radius = radius;
    }

    //...

    public static void main(String[] args){
        Ball football = new Ball("White","Taweewie",10.3);
        Ball basketball = new Ball("Orange", "Lovely", 45.7);
    }
}
```

Java access modifiers

```java
public class Ball {
    private String color;
    private String name;
    private double radius;

    public Ball(String color, String name, double radius){
        this.color = color;
        this.name = name;
        this.radius = radius;
    }

    //...

    public static void main(String[] args){
        Ball football = new Ball("White","Taweewie",10.3);
        Ball basketball = new Ball("Orange", "Lovely", 45.7);
    }
}
```

An access modifier

- The access modifier control is used to set the access level for attributes, methods across classes.
- There are 4 access modifiers in Java
  - Default
  - Private
  - Protected
  - Public
- Private and Public are commonly used.

A public access modifier

- The public access is set to a method that is accessible by method **across all classes.**
- It means that methods in different class (in same package) and methods in different package can access the public method in the package.

A private access modifier

- The private access is set to a method that is accessible by **only** methods within the **same class**