

# ADVANCED INHERITANCE

Author: Boaz Kantor

The Interdisciplinary Center, Herzliya

Introduction to Computer Science

Winter 2009-10 Semester

```

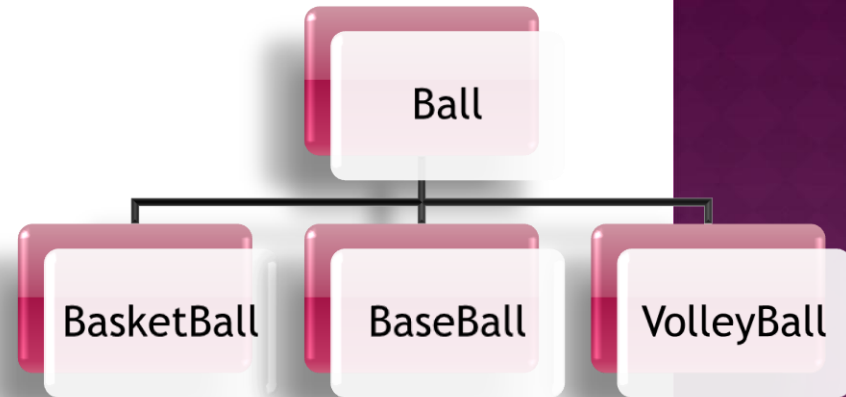
class Ball {
    private Image image;
    public void setImage(Image image) {
        this.image = image;
    }
}
class BasketBall extends Ball {
    public BasketBall() {
        setImage(new Image("

"));
    }
}
class BaseBall extends Ball {
    public BaseBall() {
        setImage (new Image ("

"));
    }
}
class VolleyBall extends Ball {
    public VolleyBall() {
        setImage (new Image ("

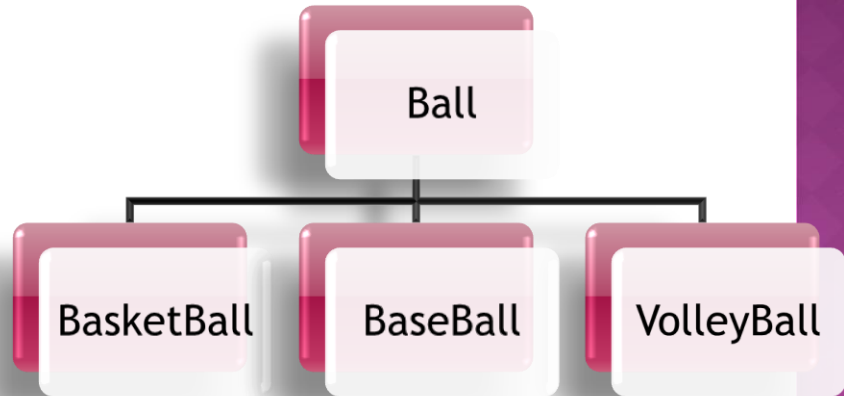
"));
    }
}

```



```
class Ball {
    private Color image;
    public void setImage(Image image) {
        this.image = image;
    }
}

class Irrelevant {
    public static void main(String[] args) {
        Ball basketball = new Basketball();
        Ball baseball = new BaseBall();
        Ball vball = new VolleyBall();
        Ball ball = new Ball();
    }
}
```



WTH?!  
(what image?)

# ABSTRACT CLASS

## ⦿ Definition:

- Cannot be instantiated
- Defined with the “abstract” keyword:

```
public abstract class Ball {
```

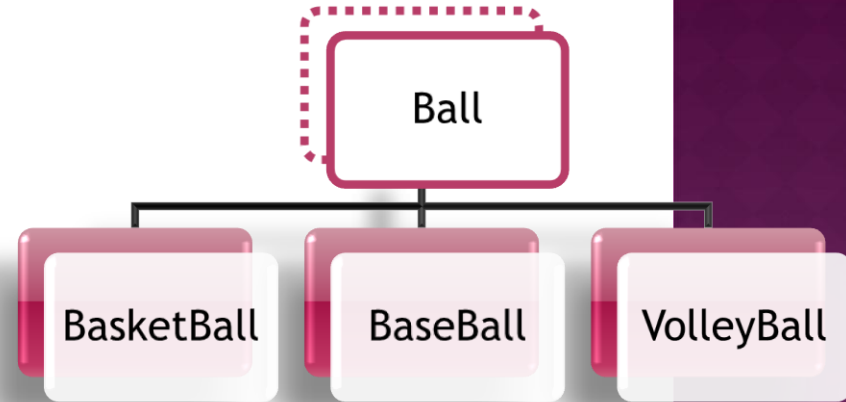
## ⦿ When to use:

- Logically can't be instantiated

```
abstract class Ball {
    private Color image;
    public void setImage(Image image) {
        this.image = image;
    }
    public int getCircumference() {
        return 0;
    }
}

class Basketball extends Ball {
    public Basketball() {
        setImage(new Image("🏀"));
    }
    public int getCircumference() {
        return 75;        // 29.5"
    }
}

class Baseball extends Ball {
    public Baseball() {
        setImage(new Image("⚾"));
    }
    public int getCircumference() {
        return 23;        // 9"
    }
}
```



# ABSTRACT METHOD

## ⦿ Definition:

- Cannot be implemented
- Must be overridden (and implemented)
- Defined with the “abstract” keyword:

```
public abstract int getCircumference();
```

## ⦿ When to use:

- No default behavior

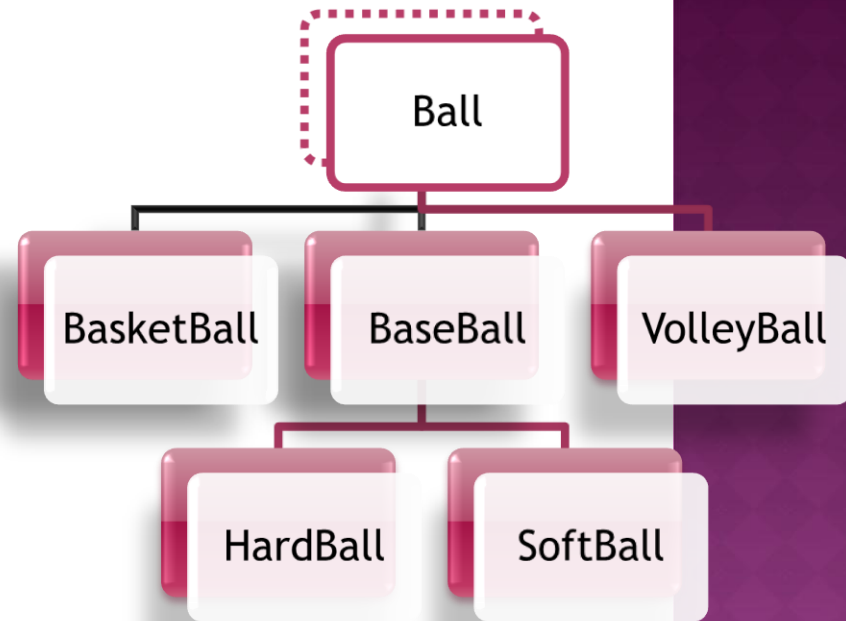
```

abstract class Ball {
    private Color image;
    public void setImage(Image image) {
        this.image = image;
    }
    public abstract int getCircumference();
}

class Basketball extends Ball {
    public Basketball() {
        setImage(new Image("🏀"));
    }
    public int getCircumference() {
        return 75;           // 29.5"
    }
}

class BaseBall extends Ball {
    public BaseBall() {
        setImage (new Image("⚾"));
    }
    // 9" for hardball, 12" for softball
    public abstract int getCircumference();
}

```



# ABSTRACT METHOD & CLASS

- If a class has an abstract method, it must be defined as an abstract class
- Makes sense:
  - What implementation would an object have in its unimplemented abstract method?
  - Therefore we forbid instantiation

```

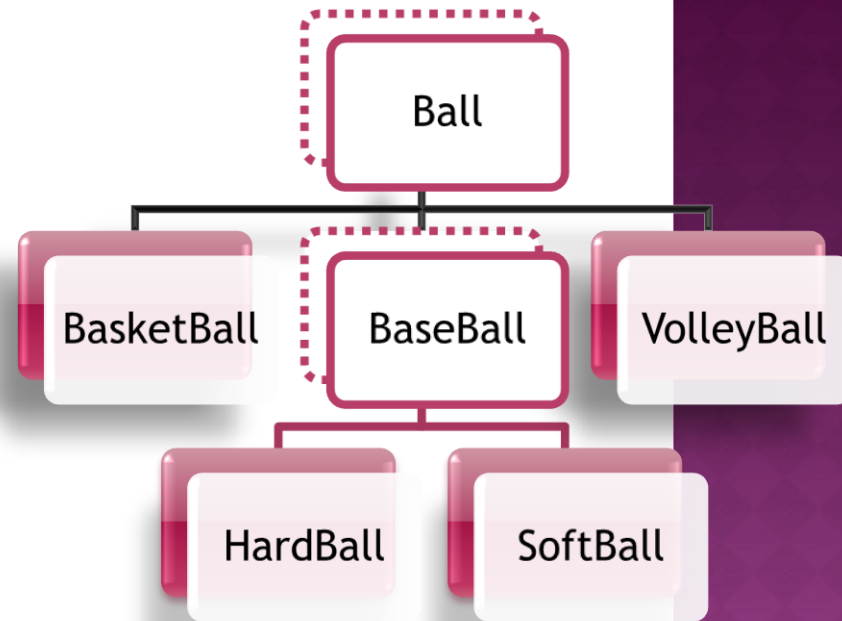
abstract class Ball {
    private Color image;
    public void setImage(Image image) {
        this.image = image;
    }
    public abstract int getCircumference();
}

abstract class BaseBall extends Ball {
    public BaseBall() {
        setImage (new Image("🏏"));
    }
    // 9" for hardball, 12" for softball
    public abstract int getCircumference();
}

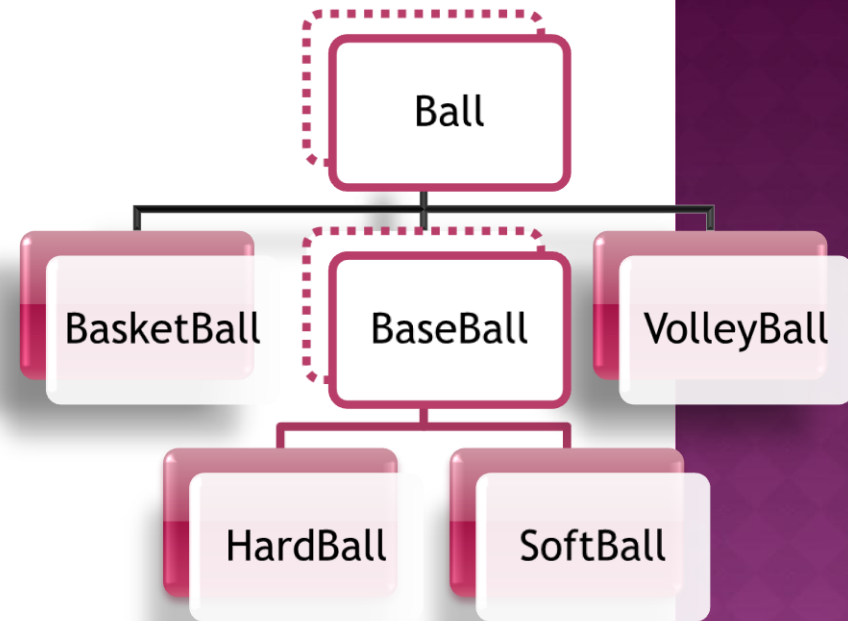
class HardBall extends BaseBall {
    public int getCircumference() {
        return 23;
    }
}

class SoftBall extends BaseBall {
    public int getCircumference() {
        return 30;
    }
}

```



```
class Irrelevant {  
    public static void main(String[] args) {  
        Ball b1 = new BasketBall();  
        Ball b2 = new BaseBall();  
        Ball b3 = new VolleyBall();  
        Ball b4 = new HardBall();  
        Ball b5 = new SoftBall();  
        Ball b6 = new Ball();  
    }  
}
```



```

abstract class Ball {
    private Color image;

    public void setImage(Image image) {this.image = image;}
    public abstract int getCircumference();
}

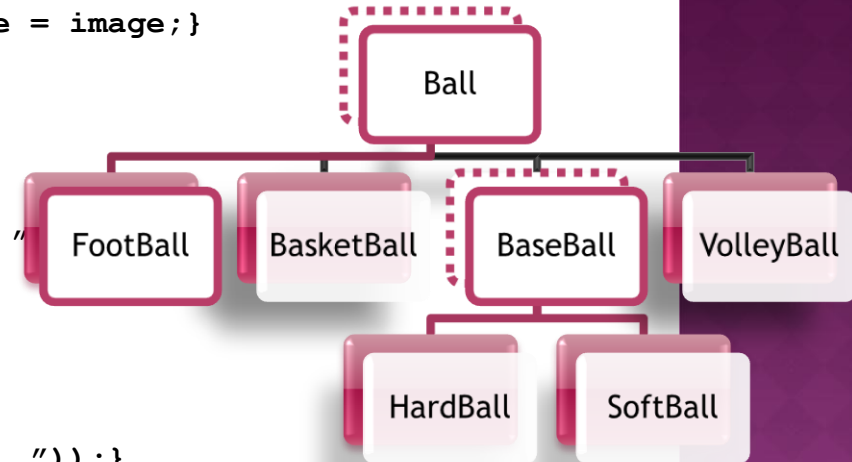
class Basketball extends Ball {
    public Basketball() {setImage(new Image("
    public int getCircumference() {return 75;}
}

class VolleyBall extends Ball {
    public VolleyBall () {setImage(new Image("
    public int getCircumference() {return 66;}
}

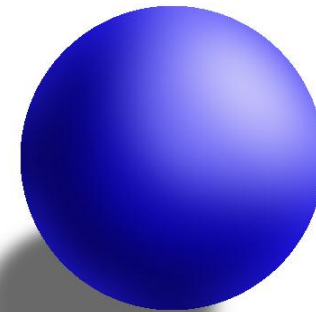
class Football extends Ball {
    public Football() {setImage (new Image("
    public int getCircumference() {
        return ???
    }
}

class Sphere {
    private int radius;
    public Sphere(int radius) {this.radius = radius;}
    public int getCircumference() {return 2*Math.PI*radius;}
}

```



");}



```

abstract class Ball {
    private Color image;
    public void setImage(Image image) {this.image = image;}
    public abstract int getCircumference();
}

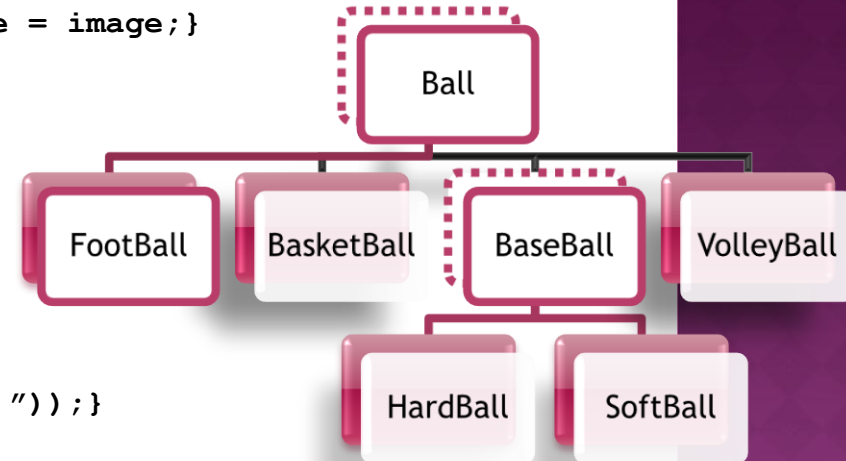
abstract class SphereShaped {
    public abstract int getCircumference();
}

class Basketball extends Ball, SphereShaped {
    public Basketball() {setImage(new Image("
    public int getCircumference() {return 75;}
}

class VolleyBall extends Ball, SphereShaped {
    public VolleyBall () {setImage(new Image("
    public int getCircumference() {return 66;}
}

class Sphere extends SphereShaped {
    private int radius;
    public Sphere(int radius) {this.radius = radius;}
    public int getCircumference() {return 2*radius;}
}

```



- ⦿ What's the problem???
- ⦿ No multiple inheritance in Java!!

# INTERFACES

## ⦿ Definition:

- Cannot be instantiated
- Defined with the “interface” keyword:

```
public interface SphereShaped {
```

## ⦿ When to use:

- Same as a class with only abstract methods
- The “able” trick:
  - Comparable, Closeable, Cloneable, ...
- Logic!
- When there’s nothing to “extend”, just to “implement”

```
abstract class Ball {
    private Color image;
    public void setImage(Image image) {this.image = image;}
}
public interface SphereShaped {
    public int getCircumference();
}
class Basketball extends Ball implements SphereShaped {
    public Basketball() {setImage(new Image("  "));}
    public int getCircumference() {return 75;}
}
class VolleyBall extends Ball implements SphereShaped {
    public VolleyBall () {setImage(new Image("  "));}
    public int getCircumference() {return 66;}
}
class Sphere implements SphereShaped {
    private int radius;
    public Sphere(int radius) {this.radius = radius;}
    public int getCircumference() {return 2*Math.PI*radius;}
}
class Football extends Ball {
    public Football() {setImage (new Image("  "));}
}
```

終

# ADVANCED INHERITANCE

Author: Boaz Kantor

The Interdisciplinary Center, Herzliya

Introduction to Computer Science

Winter 2009-10 Semester