Inheritance

Extending Classes

SoftUni Team

Technical Trainers

Table of Contents

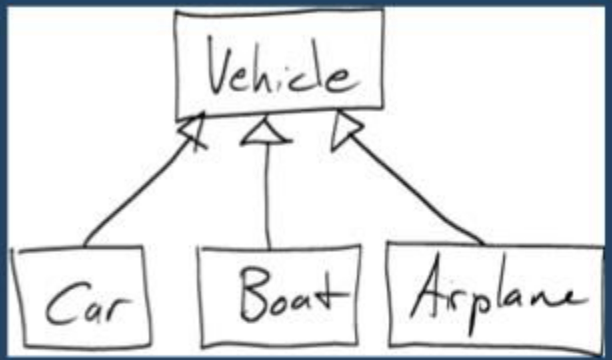
SoftUni Foundation

Have a Question?

SoftUni Foundation

sli.do

#java-advanced

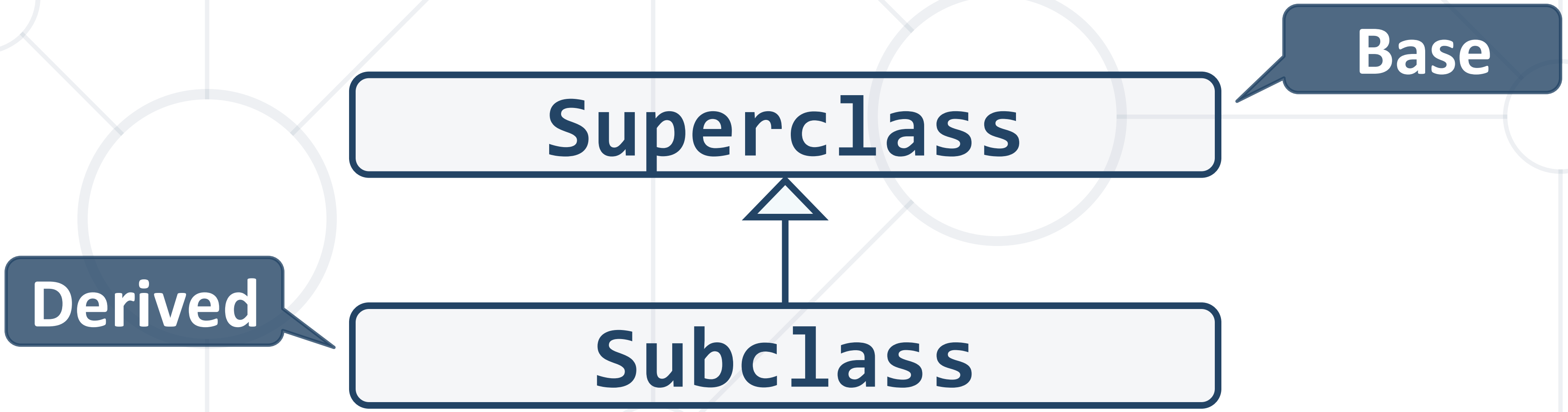


Inheritance

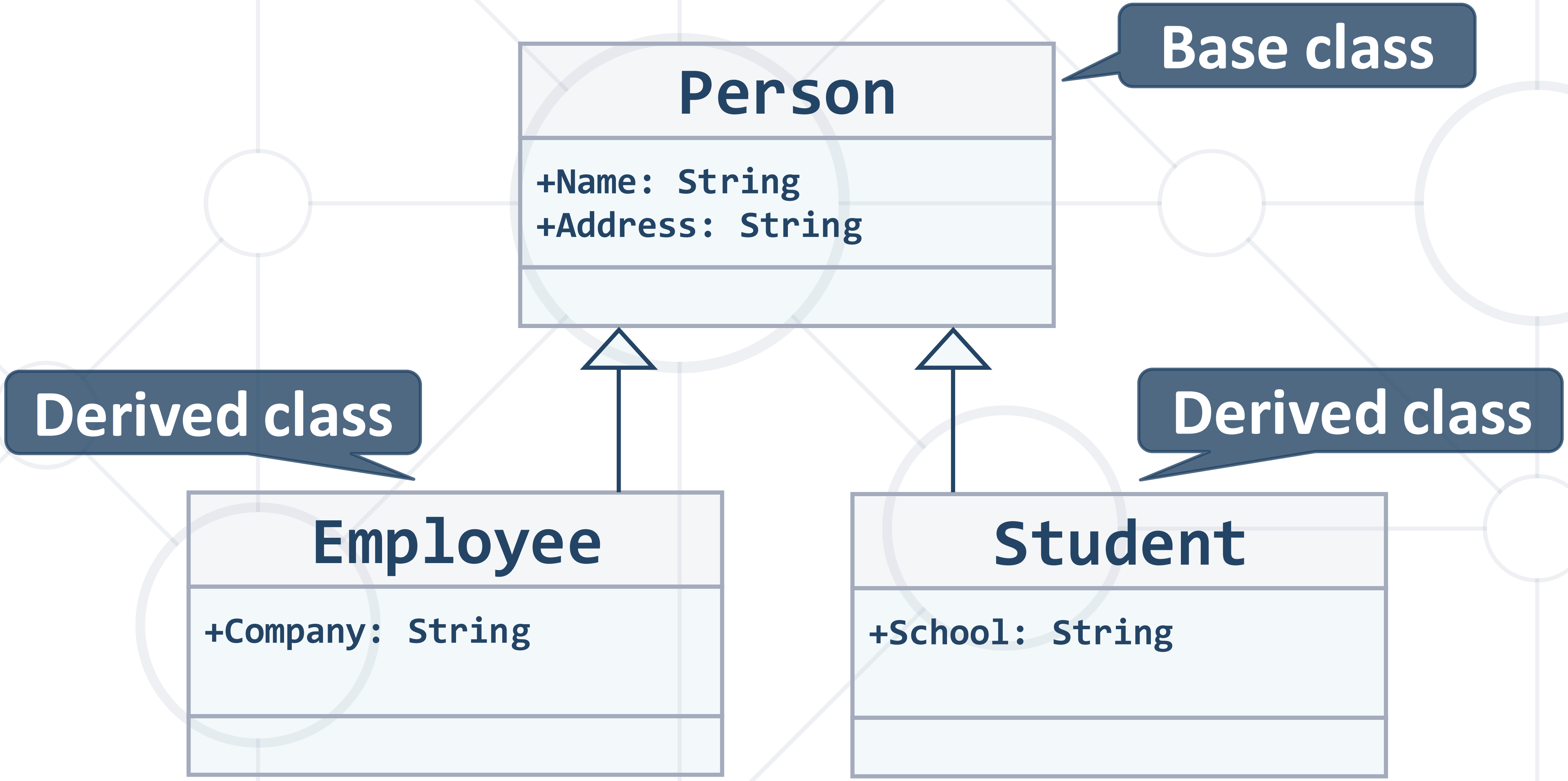
Extending Classes

Inheritance

- **Superclass** - Parent class, Base Class
  - The class giving its members to its child class
- **Subclass** - Child class, Derived Class
  - The class taking members from its base class

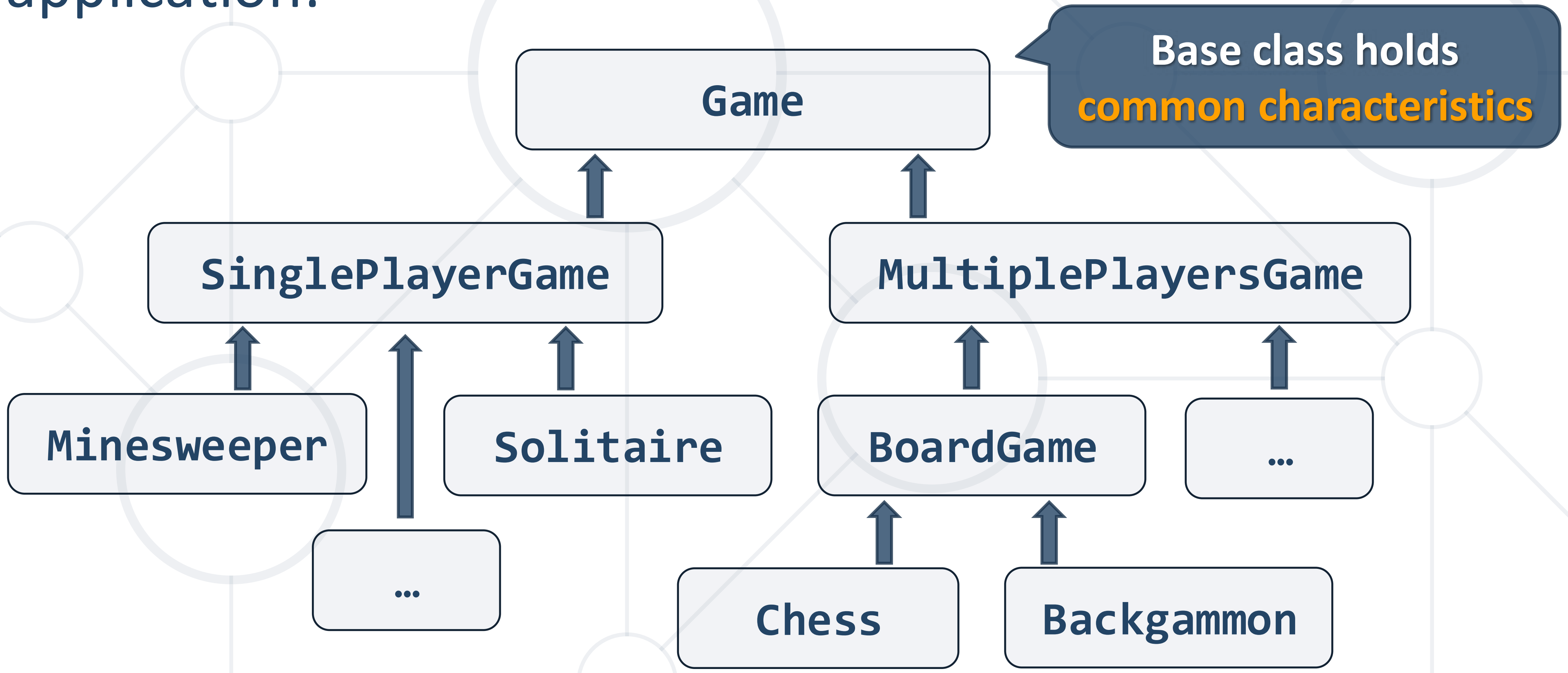


Inheritance – Example

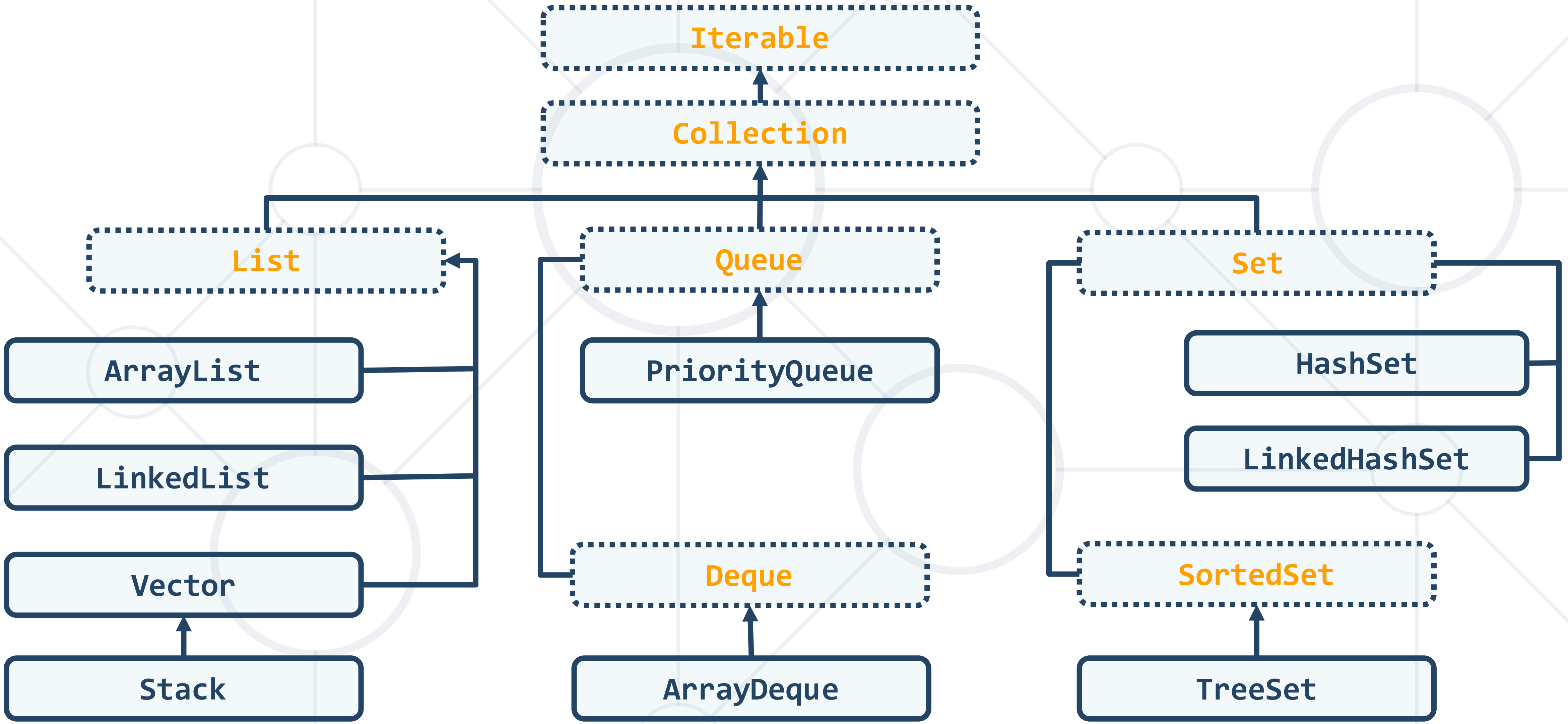


Class Hierarchies

- **Inheritance** leads to **hierarchies** of classes and/or interfaces in an application:

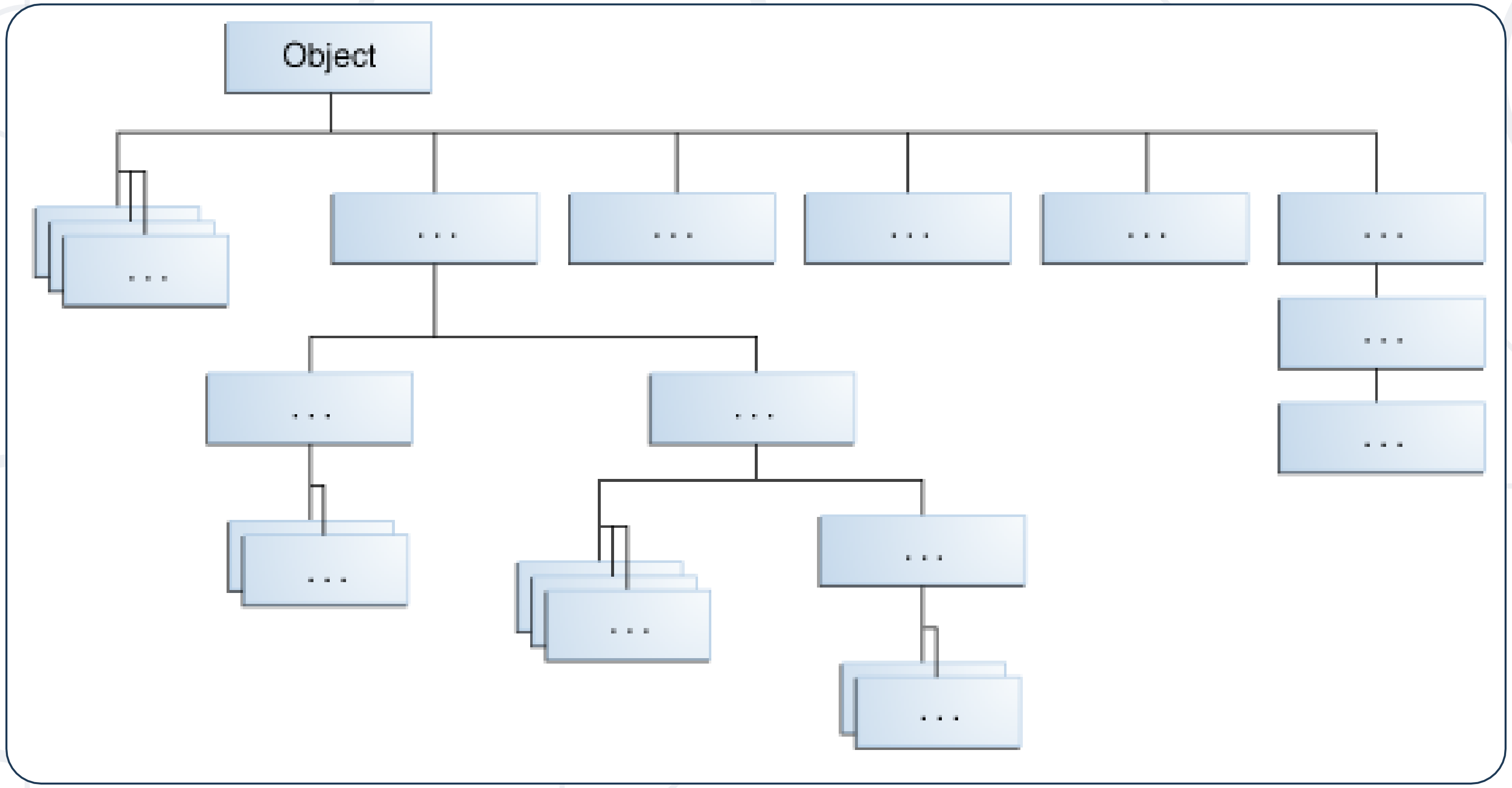


Class Hierarchies – Java Collection



Java Platform Class Hierarchy

- **Object** is at the root of Java Class Hierarchy

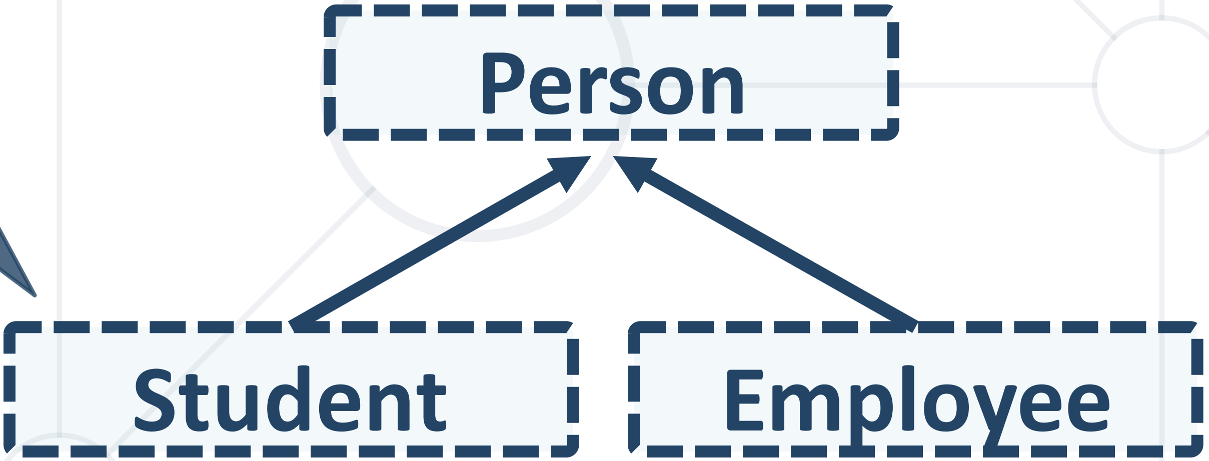


Inheritance in Java

- Java supports inheritance through **extends** keyword

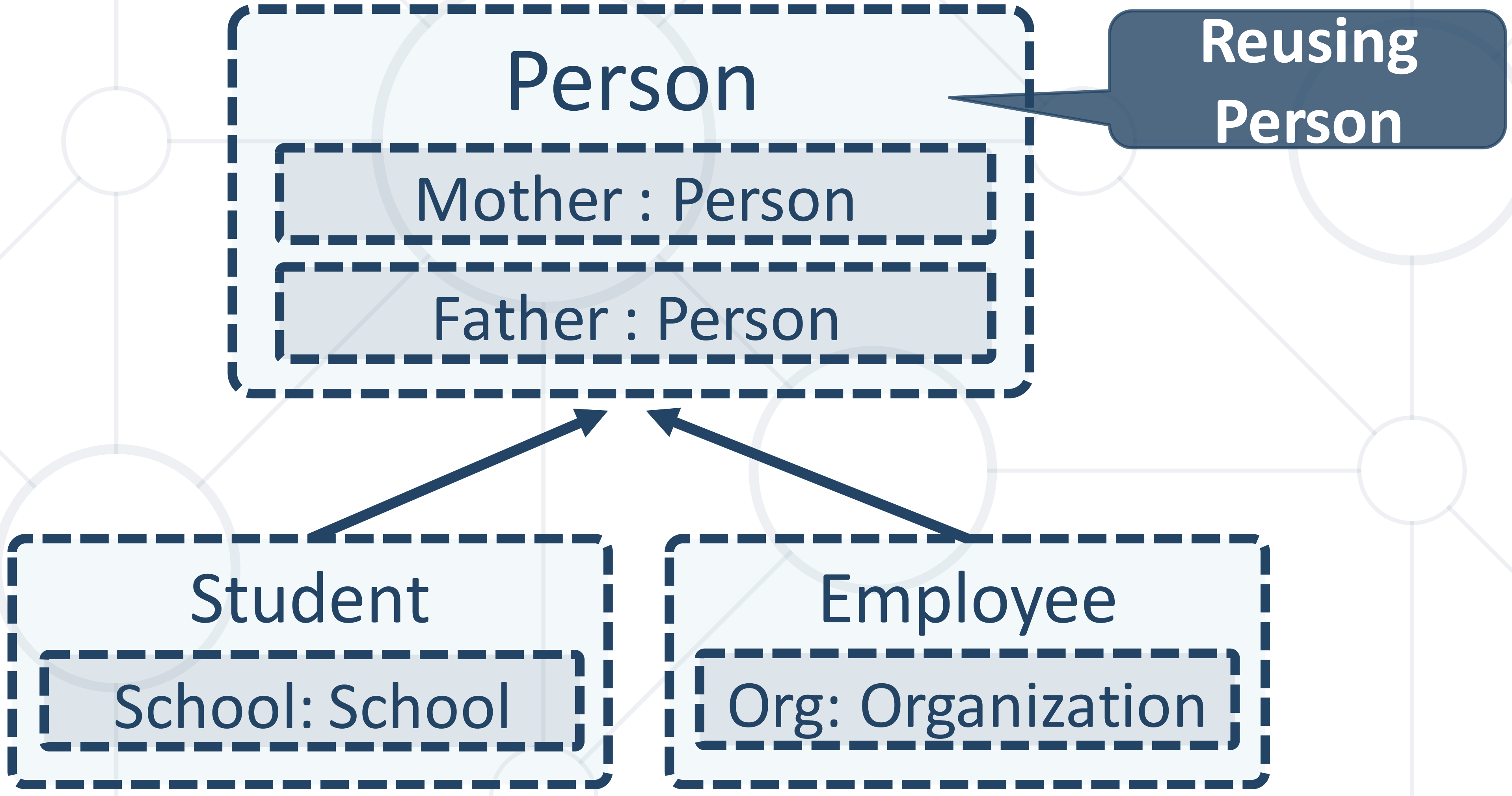
```
class Person { … }

class Student extends Person { … }
class Employee extends Person { … }
```

Student **extends** Person



Inheritance - Derived Class

- Class **takes all members** from another class



Using Inherited Members

- You can access inherited members as usual

```
class Person { public void sleep() { … } }
class Student extends Person { … }
class Employee extends Person { … }
```

```
Student student = new Student();
student.sleep();
Employee employee = new Employee();
employee.sleep();
```

Reusing Constructors

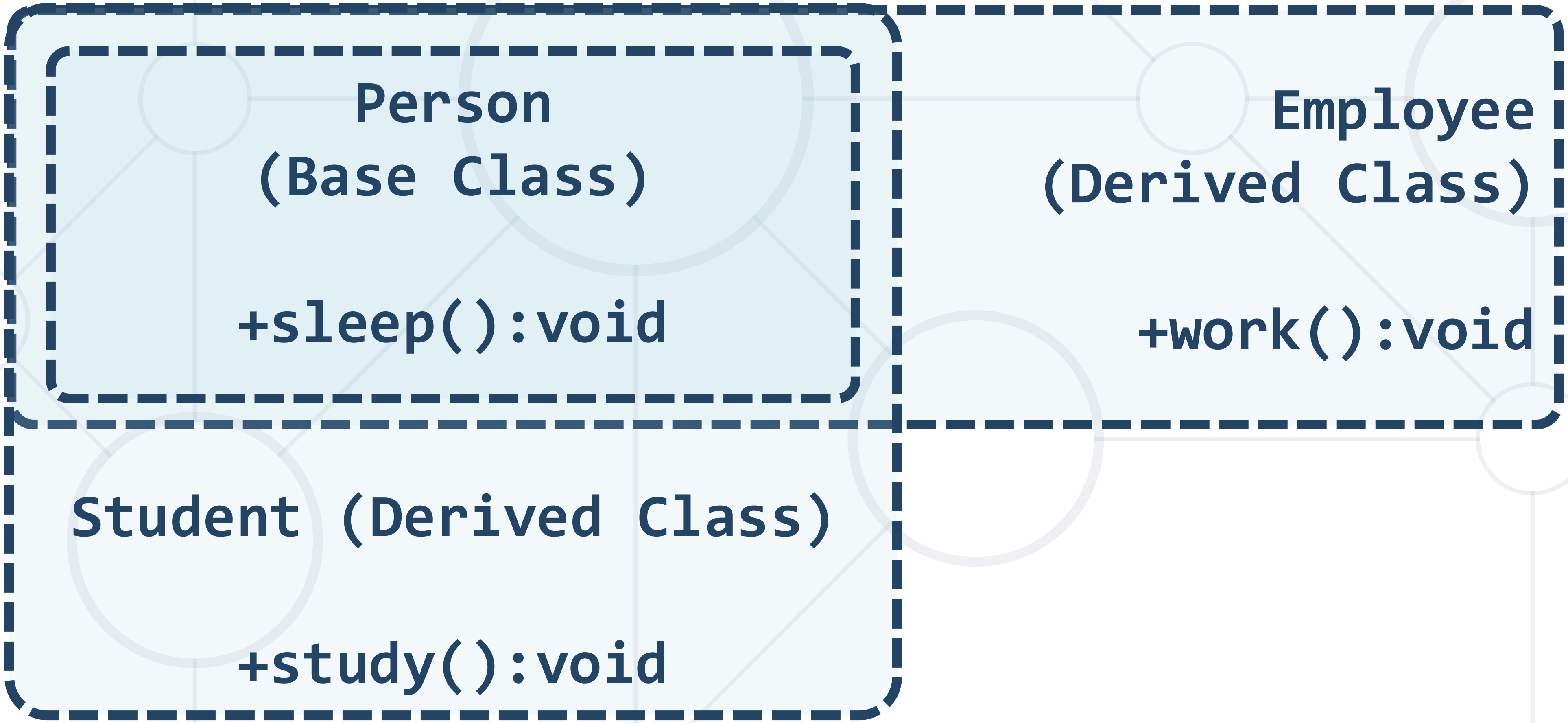
- Constructors are **not inherited**
- Constructors **can be reused** by the child classes

```
class Student extends Person {
  private School school;
  public Student(String name, School school) {
    super(name);
    this.school = school;
  }
}
```

Constructor call should be first

Thinking About Inheritance - Extends

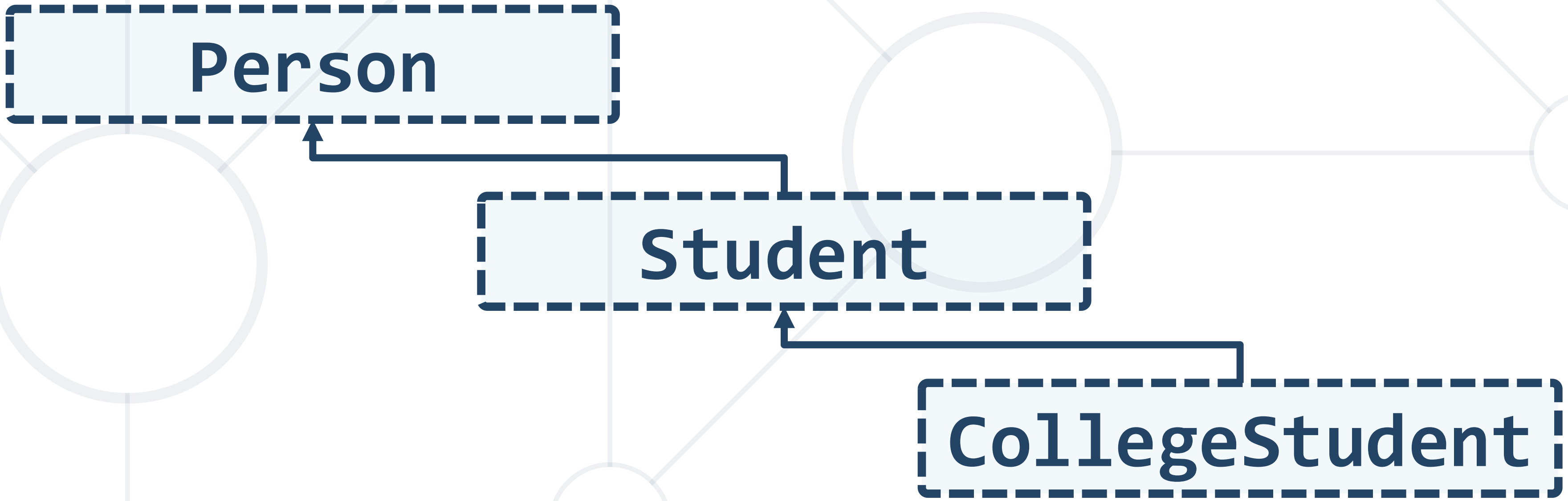
- Derived class instance **contains** instance of its base class



Inheritance

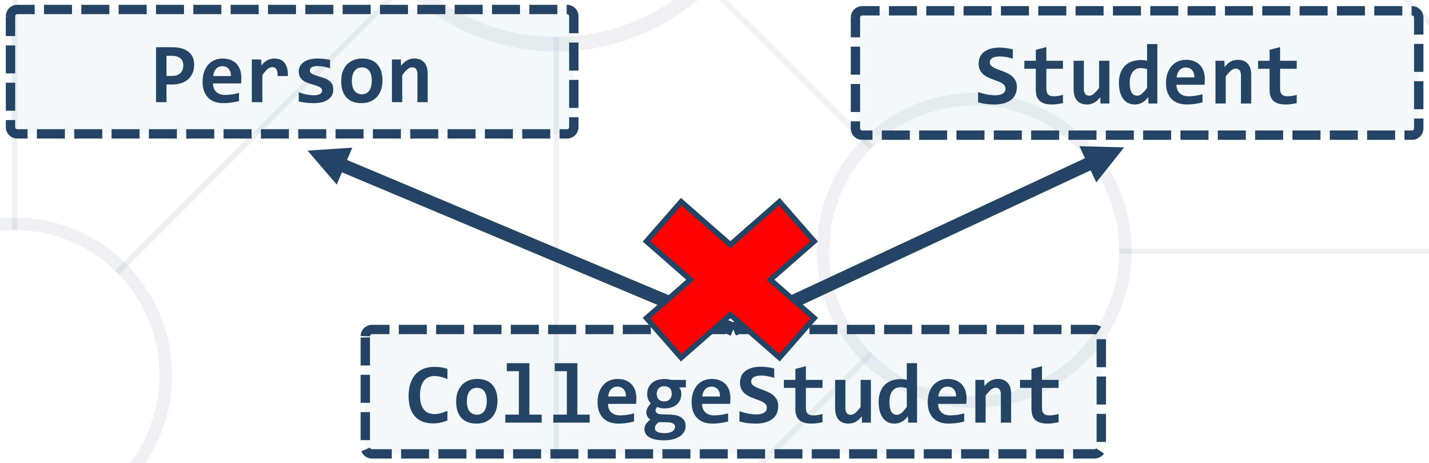
- Inheritance has a **transitive relation**

```
class Person { … }
class Student extends Person { … }
class CollegeStudent extends Student { … }
```



Multiple Inheritance

- In Java there is no **multiple** inheritance
- Only **multiple interfaces can be implemented**

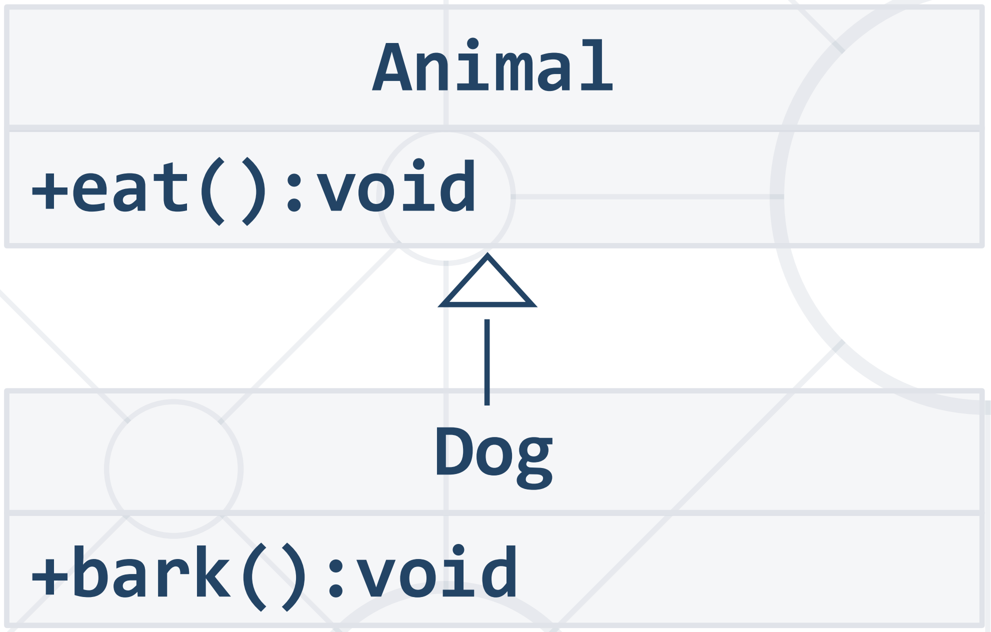


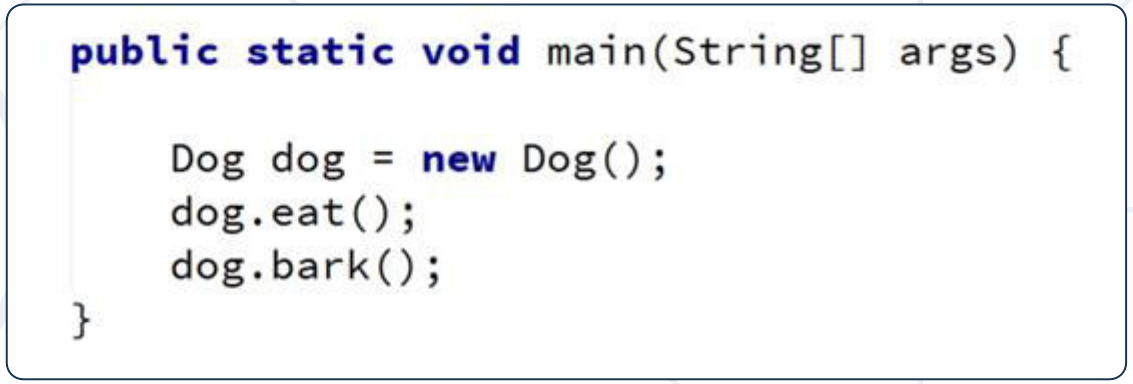
Access to Base Class Members

- Use the **super** keyword

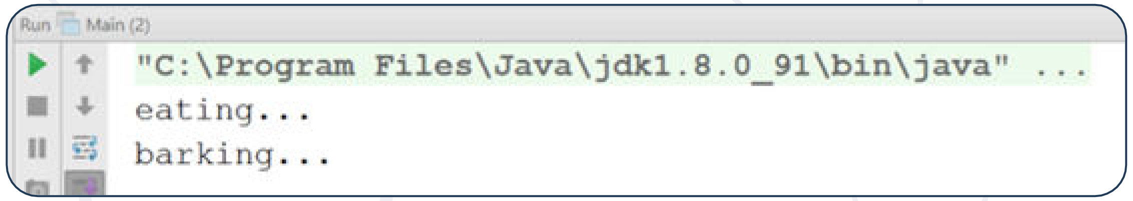
```
class Person { … }

class Employee extends Person {
  public void fire(String reasons) {
    System.out.println(
        super.name +
        " got fired because " + reasons);
  }
}
```

Problem: Single Inheritance





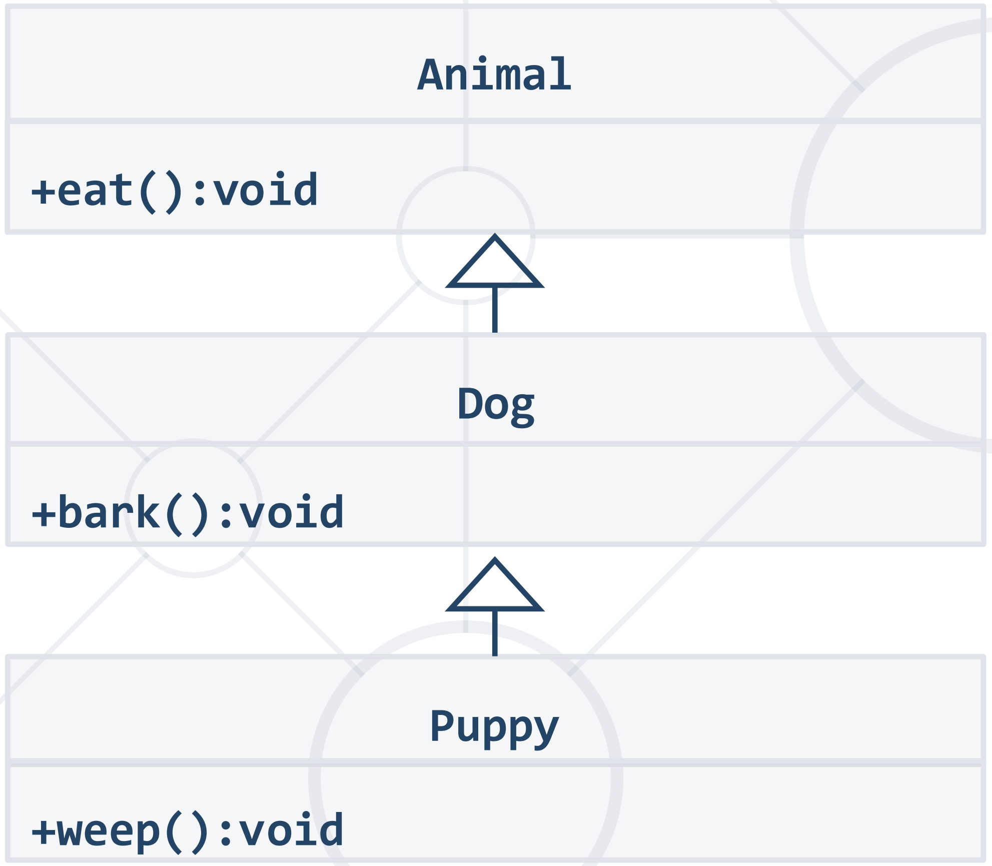
```
public static void main(String[] args) {

    Dog dog = new Dog();
    dog.eat();
    dog.bark();
}
```



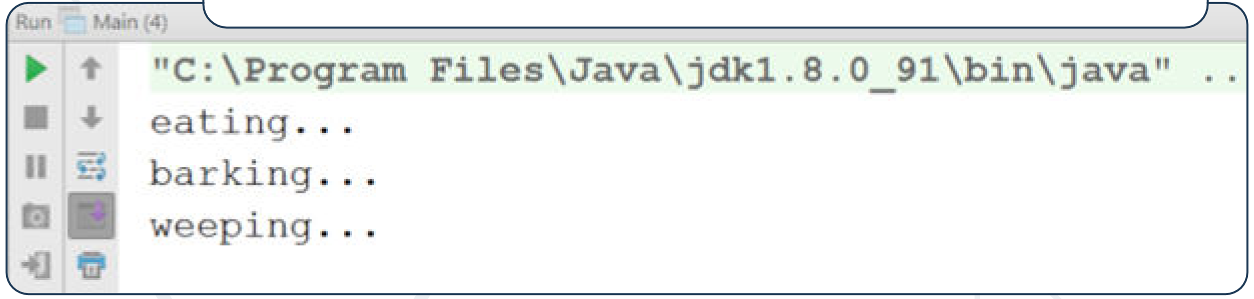
```
"C:\Program Files\Java\jdk1.8.0_91\bin\java" ...
eating...
barking...
```

Check your solution here :https://judge.softuni.bg/Contests/1574/Inheritance-Lab

Problem: Multiple Inheritance



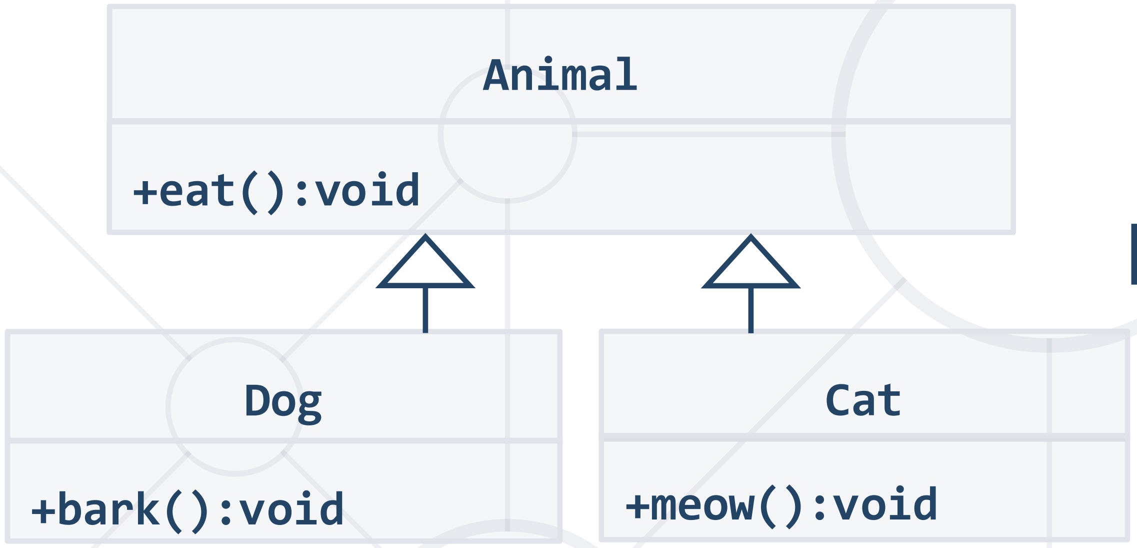
```java
Puppy puppy = new Puppy();
puppy.eat();
puppy.bark();
puppy.weep();
```



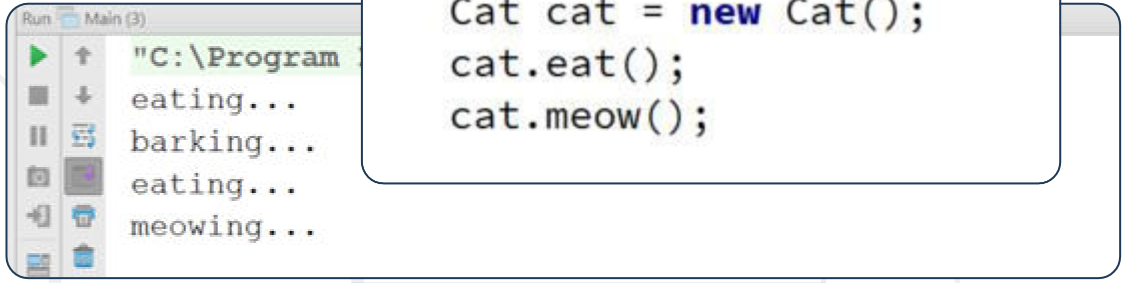
```
"C:\Program Files\Java\jdk1.8.0_91\bin\java" ..
eating...
barking...
weeping...
```

Check your solution here :https://judge.softuni.bg/Contests/1574/Inheritance-Lab

Problem: Hierarchical Inheritance



```
Dog dog = new Dog();
dog.eat();
dog.bark();

Cat cat = new Cat();
cat.eat();
cat.meow();
```



```
"C:\Program
eating...
barking...
eating...
meowing...
```

Check your solution here :https://judge.softuni.bg/Contests/1574/Inheritance-Lab

Reusing Classes

Reusing Code at Class Level

Inheritance and Access Modifiers

- Derived classes **can acces all public** and **protected** members
- Derived classes can access **default** members **if in same package**
- **Private** fields **aren't inherited** in subclasses (can't be accesssed)

```
class Person {
  protected String address;
  public void sleep();
  String name;
  private String id;
}
```

Can be accessed through other methods

Shadowing Variables

- Derived classes **can hide** superclass variables

```
class Person { protected int weight; }
```

```
class Patient extends Person {
  protected float weight;
  public void method() {
    double weight = 0.5d;
  }
}
```

hides int weight

hides both

Shadowing Variables - Access

- Use **super** and **this** to specify member access

```java
class Person { protected int weight; }
```

```java
class Patient extends Person {
  protected float weight;
  public void method() {
    double weight = 0.5d;
    this.weight = 0.6f;
    super.weight = 1;
  }
}
```

Local variable

Instance member

Base class member

Overriding Derived Methods

- A **child class** can redefine existing methods

```
public class Person {
   public void sleep() {
      System.out.println("Person sleeping"); }
}

public class Student extends Person {
   @Override
   public void sleep(){
      System.out.println("Student sleeping"); }
}
```

Method in base class **must not be final**

Signature and return type **should match**

Final Methods

- **final** – defines a method that **can't be overridden**

```
public class Animal {
  public final void eat() { … }
}
```

```
public class Dog extends Animal {

  @Override
  public void eat() {} // Error…
}
```

Final Classes

- Inheriting from a final classes is forbidden

```
public final class Animal {
  …
}
```

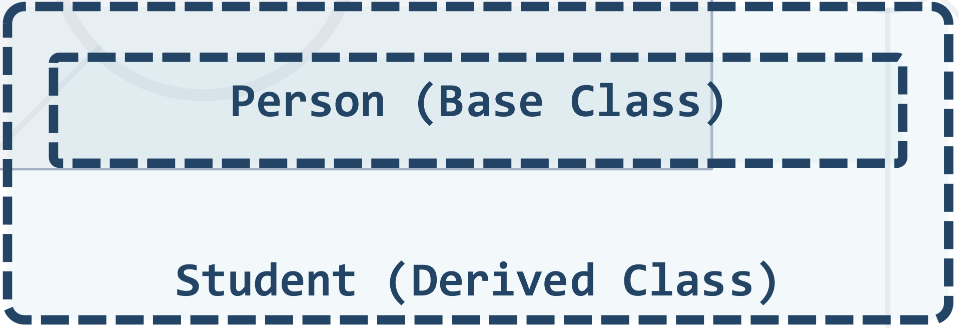
```
public class Dog extends Animal { }       // Error…
public class MyString extends String { } // Error…
public class MyMath extends Math { }      // Error…
```

Inheritance Benefits - Abstraction

- One approach for providing abstraction

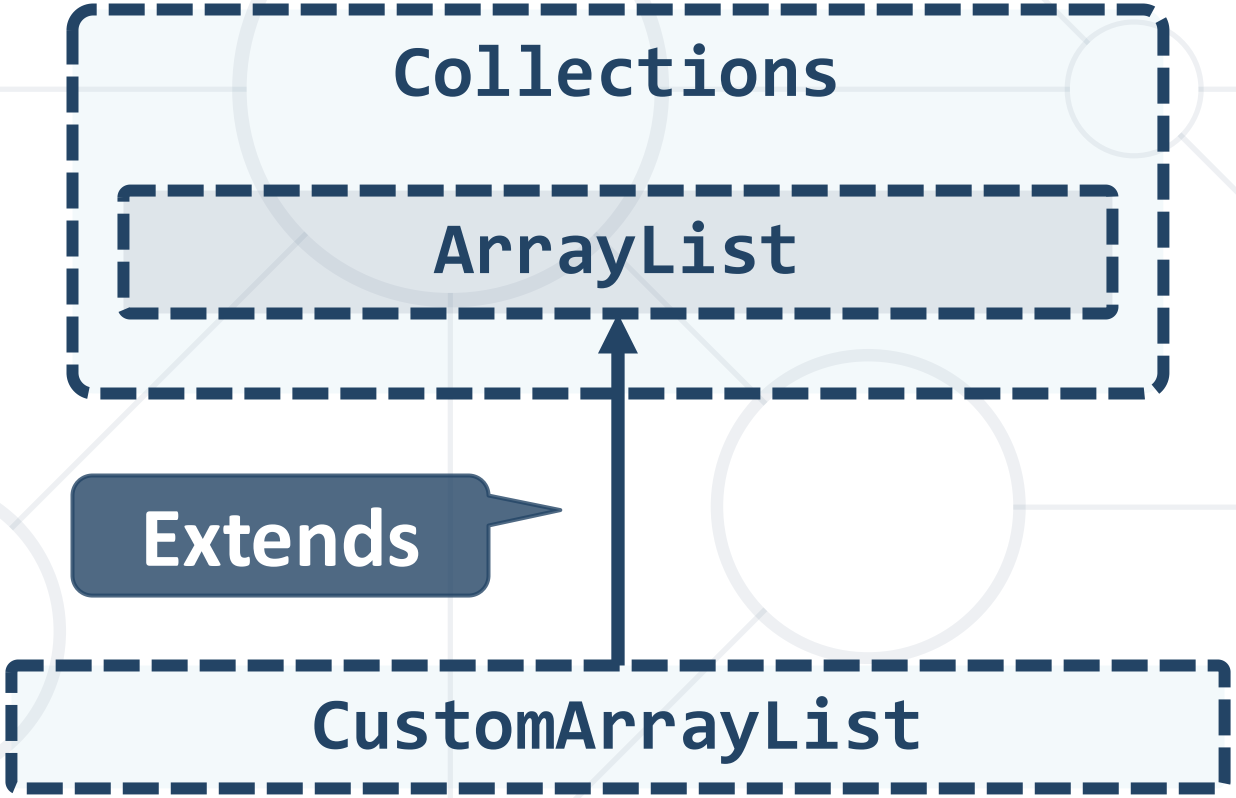
Focus on common properties

```
Person person = new Person();
Student student = new Student();

List<Person> people = new ArrayList();

people.add(person);
people.add(student);
```



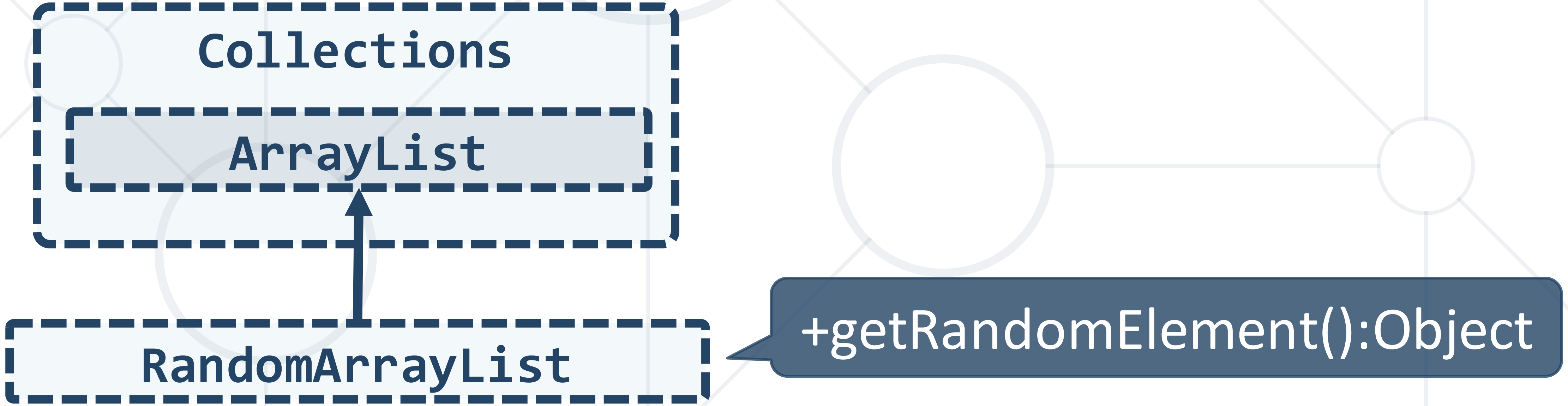
Inheritance Benefits – Extension

- We can **extend a class** that we **can't otherwise change**



Problem: Random Array List

- Create an array list that has
  - All functionality of an ArrayList
  - Function that returns and removes a random element



Check your solution here :https://judge.softuni.bg/Contests/1574/Inheritance-Lab

Solution: Random Array List

```java
public class RandomArrayList extends ArrayList {
  private Random rnd; // Initialize this…

  public Object getRandomElement() {
    int index = this.rnd.nextInt(super.size());
    Object element = super.get(index);
    super.remove(index);
    return element;
  }
}
```

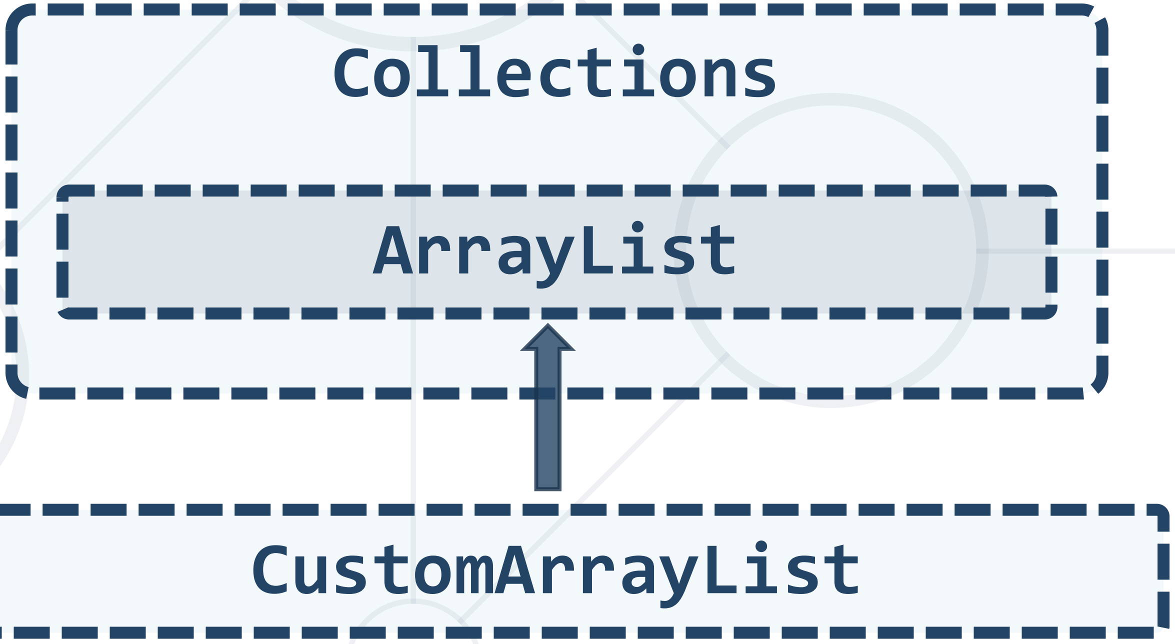
Check your solution here :https://judge.softuni.bg/Contests/1574/Inheritance-Lab

Types of Class Reuse

Extension, Composition, Delegation

Extension

- **Duplicate code** is error prone
- **Reuse classes** through **extension**
- Sometimes the only way

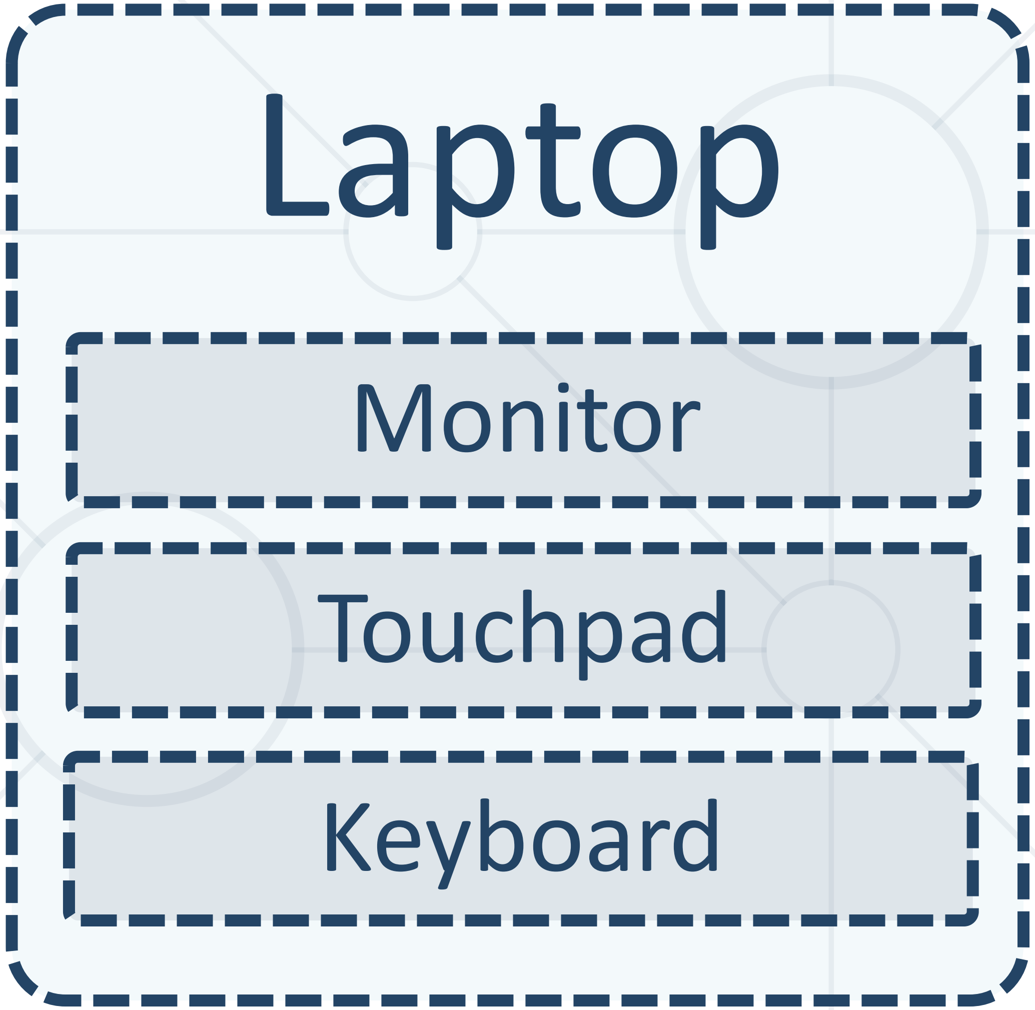


Composition

- Using classes to define classes

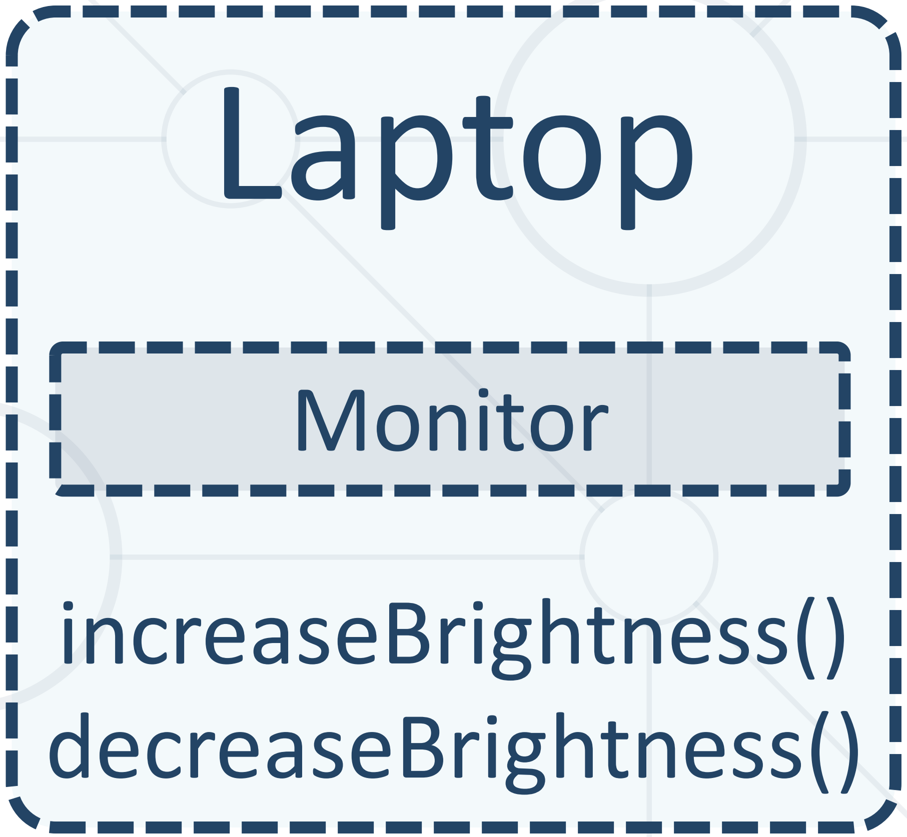
```
class Laptop {
  Monitor monitor;
  Touchpad touchpad;
  Keyboard keyboard;
  …
}
```

Reusing classes



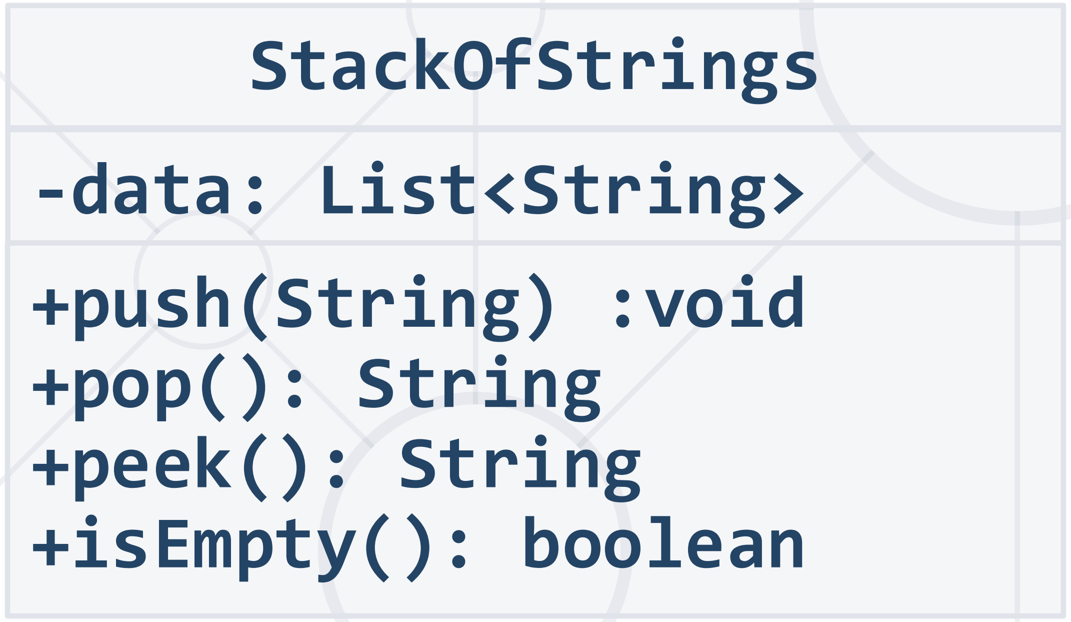
Delegation

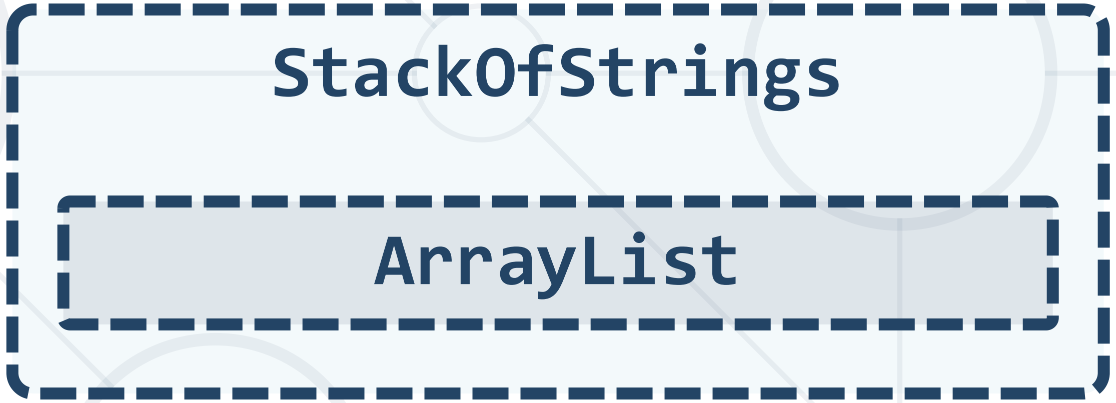
```
class Laptop {
  Monitor monitor;
  void incrBrightness() {
    monitor.brighten();
  }

  void decrBrightness() {
    monitor.dim();
  }
}
```



Problem: Stack of Strings

- Create a simple Stack class which can store only strings





```
StackOfStrings sos = new StackOfStrings();
sos.push("one");
System.out.println(sos.pop());
System.out.println(sos.isEmpty());
System.out.println(sos.peek());
```

Check your solution here :https://judge.softuni.bg/Contests/1574/Inheritance-Lab

Solution: Stack of Strings

```java
public class StackOfStrings {
  private List<String> container;
  // TODO: Create a constructor
  public void push(String item) { this.container.add(item); }
  public String pop() {
    // TODO: Validate if list is not empty
    return this.container.remove(this.container.size() - 1);
  }
}
```

Check your solution here :https://judge.softuni.bg/Contests/1574/Inheritance-Lab

When to Use Inheritance

- Classes share **IS-A** relationship ← **Too simplistic**
- Derived class **IS-A-SUBSTITUTE** for the base class
- Share the **same role**
- Derived class is the **same as the base class** but adds a **little bit more functionality**

Summary

- Inheritance is a powerful tool for **code reuse**
- **Subclass inherits** members from **Superclass**
- Subclass can **override** methods
- Look for classes with the **same role**
- Look for **IS-A** and **IS-A-SUBSTITUTE** for relationship
- Consider **Composition** and **Delegation** instead

Questions?

SoftUni

Software University

SoftUni Svetlina

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

SoftUni Diamond Partners

SoftUni Foundation

XSsoftware

SBTech we know sports

telenor

SoftwareGroup doing it right

NETPEAK

SmartIT

Postbank Решения за твоето утре

SUPER HOSTING .BG

INDEAVR Serving the high achievers

INFRAGISTICS

STEMO Computer Systems & Software

SoftUni Organizational Partners

SoftUni Foundation

ИНФОРМАЦИОННО ОБСЛУЖВАНЕ

OneBit SOFTWARE

Lukanet.com

WORLD OF MYTHS

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education and Employment Opportunities
  - softuni.bg
- Software University Foundation
  - http://softuni.foundation/
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

License