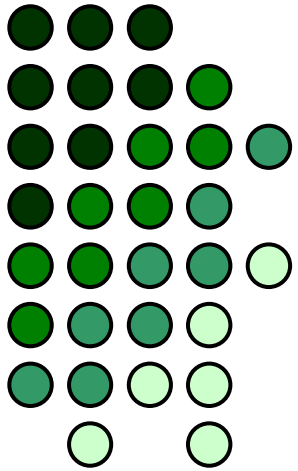
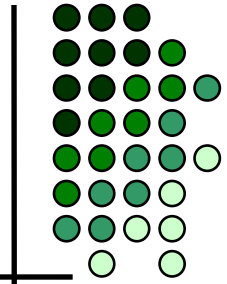
Inheritance in Java

Paul Inventado

De La Salle University



Inheritance in Java

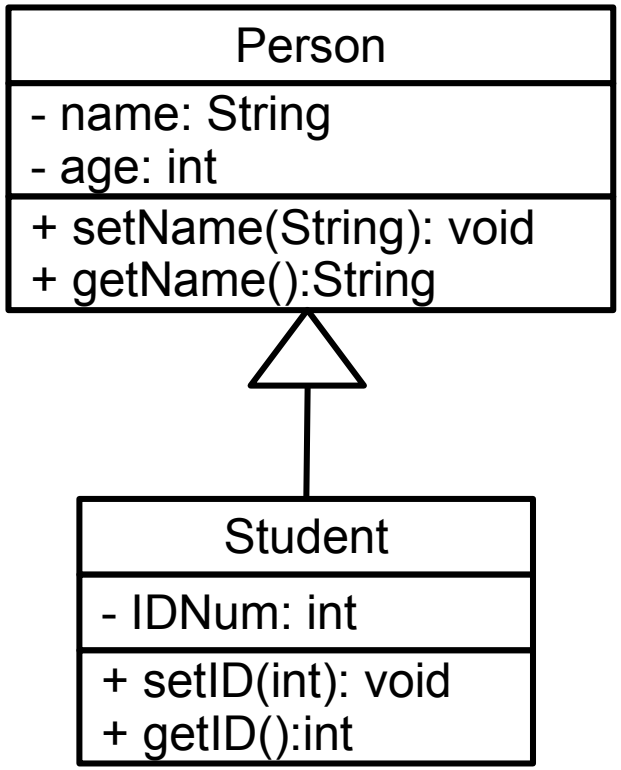


Review on Class Inheritance

- Java allows classes to extend other classes
  - i.e. **sub classes extend parent classes**
- Sub classes inherit all attributes and methods of the parent classes

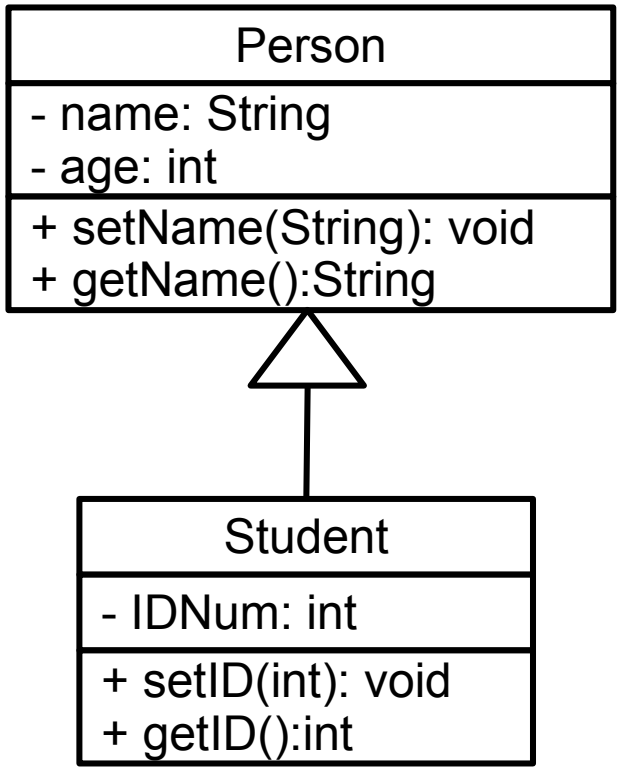
Inheritance in Java

```
public class Person
{
    private String name;
    private int age;
    public void setName(String name)
    {
        this.name=name;
    }
    public String getName()
    {
        return name;
    }
}
```



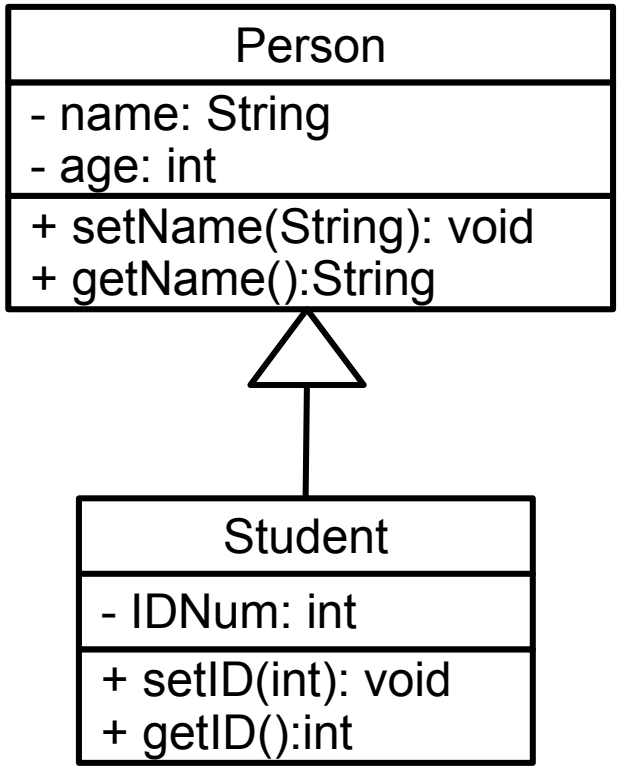
Inheritance in Java

```
public class Student extends Person
{
    private int IDNum;
    public void setID(int IDNum)
    {
        this.IDNum=IDNum;
    }
    public void getID()
    {
        return IDNum;
    }
}
```

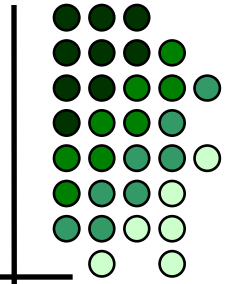


Inheritance in Java

```
public class Driver
{
    public static void main(String[] args)
    {
        Person p=new Person();
        p.setName("George");
        Student s=new Student();
        s.setName("Ringgo");
        s.setID(10105476);
    }
}
```

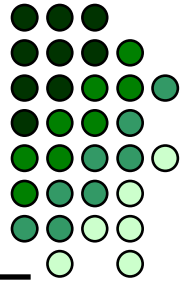


Inheritance in Java



- Unlike methods, constructors are not inherited
- Subclasses must have their own implementation of constructors

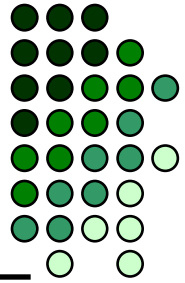
Inheritance in Java



```
public class Person
{
    private String name;
    public Person(String name)
    {
        setName(name);
    }
    public void setName(String
        name)
    {
        this.name=name;
    }
}
```

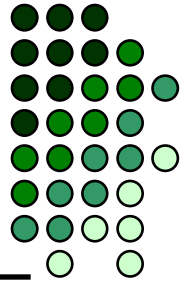
```
public class Student extends Person
{
    private int ID;
    public Student(String name)
    {
        this(name,000000);
    }
    public Student(String name, int
        ID)
    {
        setName(name);
        this.ID=ID;
    }
}
```

Inheritance in Java



- Java provides the ***super*** keyword which can be used to refer to the parent class
- It is similar to the ***this*** keyword in functionality
- The ***super*** keyword can be used to call a parent class' constructor or refer to the attributes and methods of the parent class

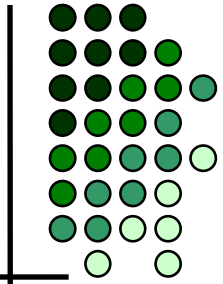
Inheritance in Java



```
public class Person
{
    private String name;
    public Person(String name)
    {
        setName(name);
    }
    public void setName(String
        name)
    {
        this.name=name;
    }
}
```

```
public class Student extends Person
{
    private int ID;
    public Student(String name)
    {
        super(name);
    }
    public Student(String name, int
        ID)
    {
        super.setName(name);
        this.ID=ID;
    }
}
```

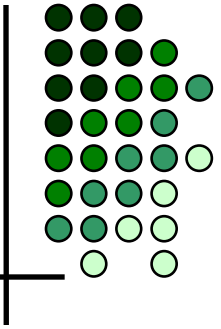
Inheritance in Java



Method Overriding

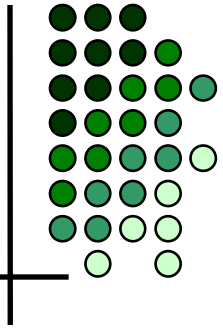
- A subclass can redefine a superclass' method by using the same signature
- When that method is called in the subclass, the subclass' version is automatically called.

Inheritance in Java



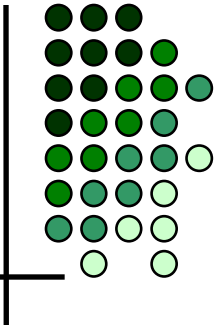
```
public class RegularEmployee
{
    int nYearsWorking;
    public Employee(int nYearsWorking)
    {
        this.nYearsWorking=nYearsWorking;
    }
    public double getSalary()
    {
         return nYearsWorking*1000;
    }
}
```

Inheritance in Java



```
public class Manager extends RegularEmployee
{
    private int nEmployees;
    public Manager(int nYearsWorking)
    {
      super(nYearsWorking);
      nEmployees=5;
    }
    public double getSalary()
    {
      return super.getSalary()*nEmployees;
    }
}
```

Inheritance in Java



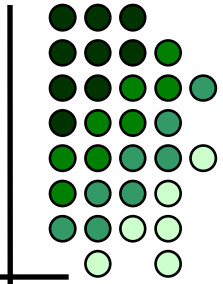
```
public class Driver
{
    public static void main(String[] args)
    {
      Employee emp1=new Employee(5);
      Manager mngr1=new Manager(5);
      System.out.println("Employee: "+emp1.getSalary());
      System.out.println("Manager: "+mngr1.getSalary());
    }
}
```

Screen Output:

Employee: 5000

Manager: 25000

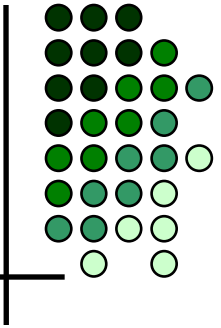
Inheritance in Java



Method Overloading

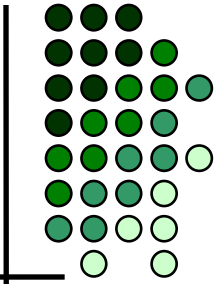
- Defining methods having the same name multiple times
- These definitions differ only on their parameters

Inheritance in Java



```
public class Student
{
    float fGrade;
    public void assignGrade(float fGrade)
    {
      this.fGrade=fGrade;
    }
    public void assignGrade(char cGrade)
    {
      switch(cGrade)
      {
            case ‘A’: fGrade=4.0; break;
            case ‘B’: fGrade=3.5; break;
            case ‘C’: fGrade=3.0; break;
            case ‘D’: fGrade=2.5; break;
      }
    }
    public float getGrade(){ return fGrade; }
}
```

Inheritance in Java

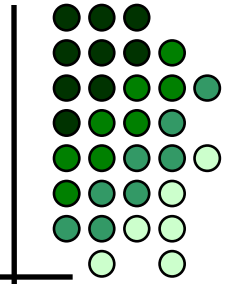


```
public class Test
{
    public static void main(String[] args)
    {
     Student s1=new Student();
     s1.assignGrade(1.0);
     System.out.println("Grade1: "+s1.getGrade());
     s1.assignGrade('B');
     System.out.println("Grade2: "+s1.getGrade());
    }
}
```

Screen Output:

```
Grade 1: 1.0
Grade 2: 3.5
```

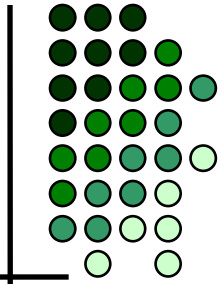
Inheritance in Java



Similarity of Method Overloading and Overriding

- Multiple methods having the same name

Inheritance in Java



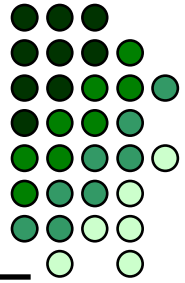
Overloading

- Different types, number or order of parameters
- Can overload within the class or from superclasses

Overriding

- Same types, number, and order of parameters
- Can override methods that are in the superclasses
- Should have same return type as overridden method
- Access modifier cannot be more limiting than the overridden method

Inheritance in Java

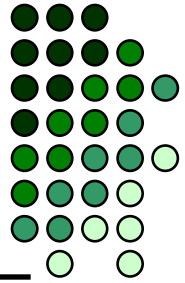


```
public class banana
{
    public void eat()
    {
      System.out.println(“Peel”);
    System.out.println(“Eat”);
    }
}
```

```
public class bananaQ extends
    banana
{
    public void eat()
    {
      System.out.println(“Peel”);
      System.out.println(“Cook”);
      System.out.println(“Eat”);
    }
}
```

overriding

Inheritance in Java

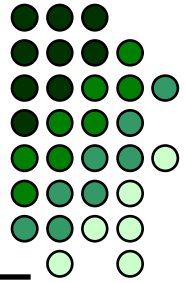


```
public class kamote
{
    public void eat()
    {
      System.out.println("Cook");
      System.out.println("Eat");
    }
}
```

overloading

```
public class kamoteQ
{
    private String sColor;
    private float fQuality;

    public void setInfo(String sColor)
    {
        this.sColor=sColor;
    }
    public void setInfo(String sColor,
        float fQuality)
    {
        setInfo(sColor);
        this.fQuality=fQuality;
    }
}
```

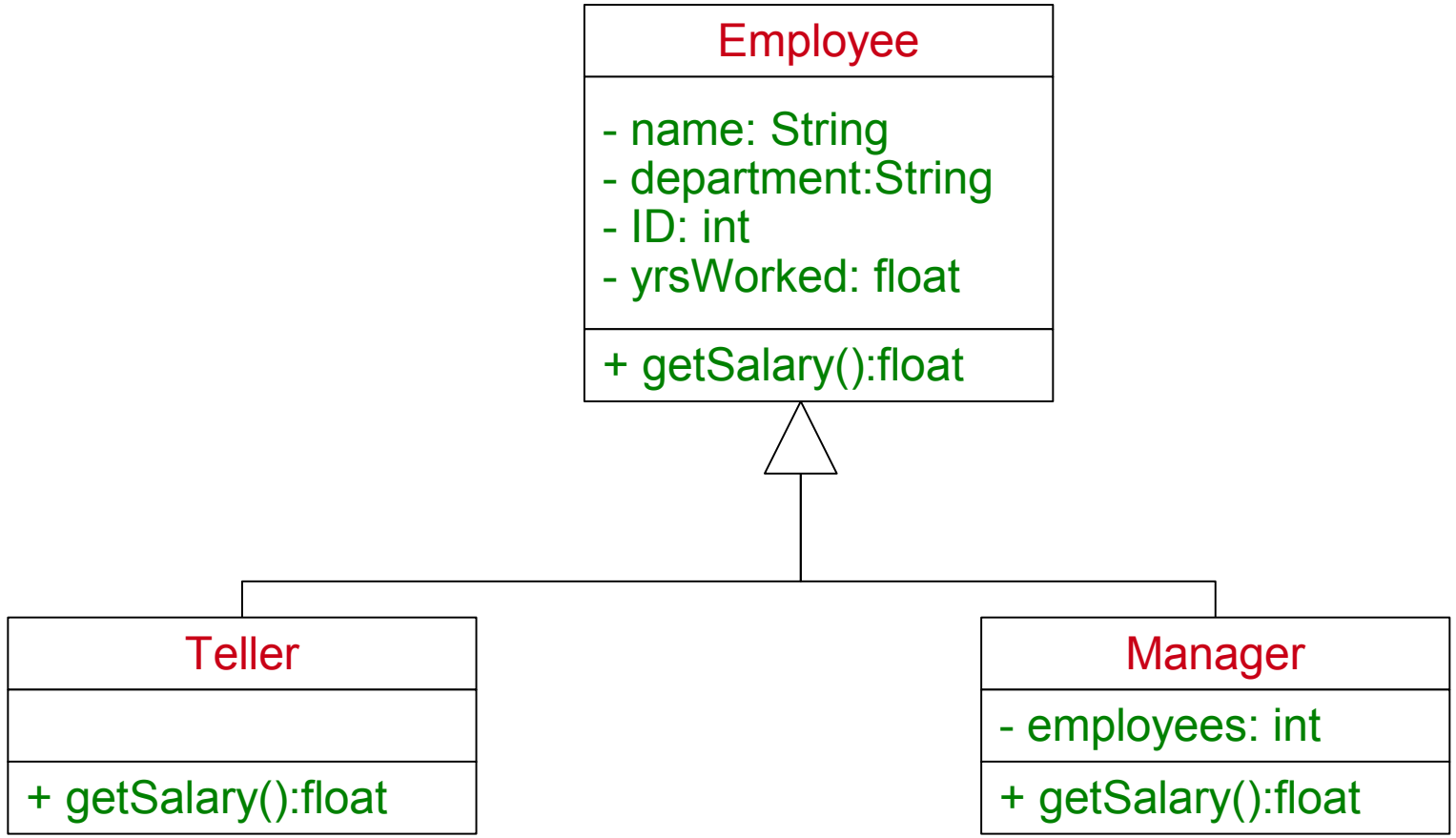
Inheritance in Java



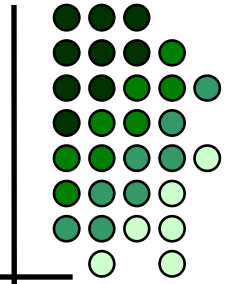
```
public watermelon
{
    public void sing()
    {
      System.out.println("Watermelon … akong … nais
    malaman … maaari bang magtanong?");
    }
}

public watermelonQ extends watermelon
{
    public void sing(int nRepetition)
    {
      for(int i=0;i<nRepetition;i++)
            sing();
    }
}
```

overloading

Inheritance in Java

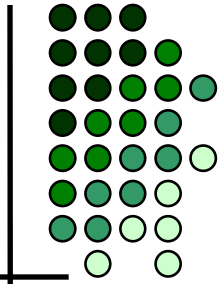


Inheritance in Java



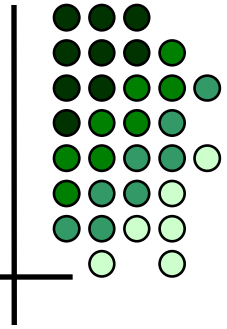
- The salary of an employee is the number of years worked * 500;
- The salary of the teller is calculated as: number of years worked * 1000.
- The salary of the manager is calculated as: number of years work * number of employees * 500.
- Create a **Display** class which will have the ***displayEmployeeSalary()*** method. Implement this method such that it will print out the correct salary of a manager or a teller object.

Inheritance in Java



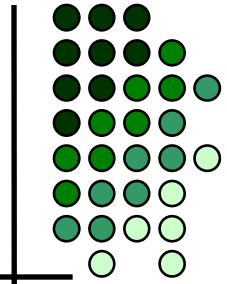
```
public class Employee
{
  String name;
  int id;
  String department;
  int yrsWorked;

  public int getSalary()
  {
      return yrsWorked*500;
  }
  //get and set functions

}
```

Inheritance in Java



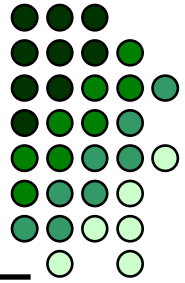
```
public class Teller extends Employee
{
  public int getSalary()
  {
    return yrsWorked*1000;
  }
}
```

Inheritance in Java



```
public class Manager extends Employee
{
  int employees;
  public int getSalary()
  {
    return yrsWorked*employees*500;
  }
}
```

Inheritance in Java



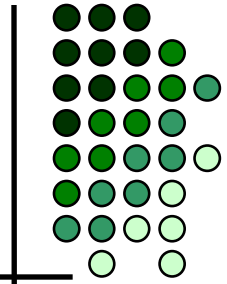
Displaying employee salary:

- What approach should be used?
- Overloading

```
public void displayEmployeeSalary(Manager m){…}

public void displayEmployeeSalary(Teller t){…}
```

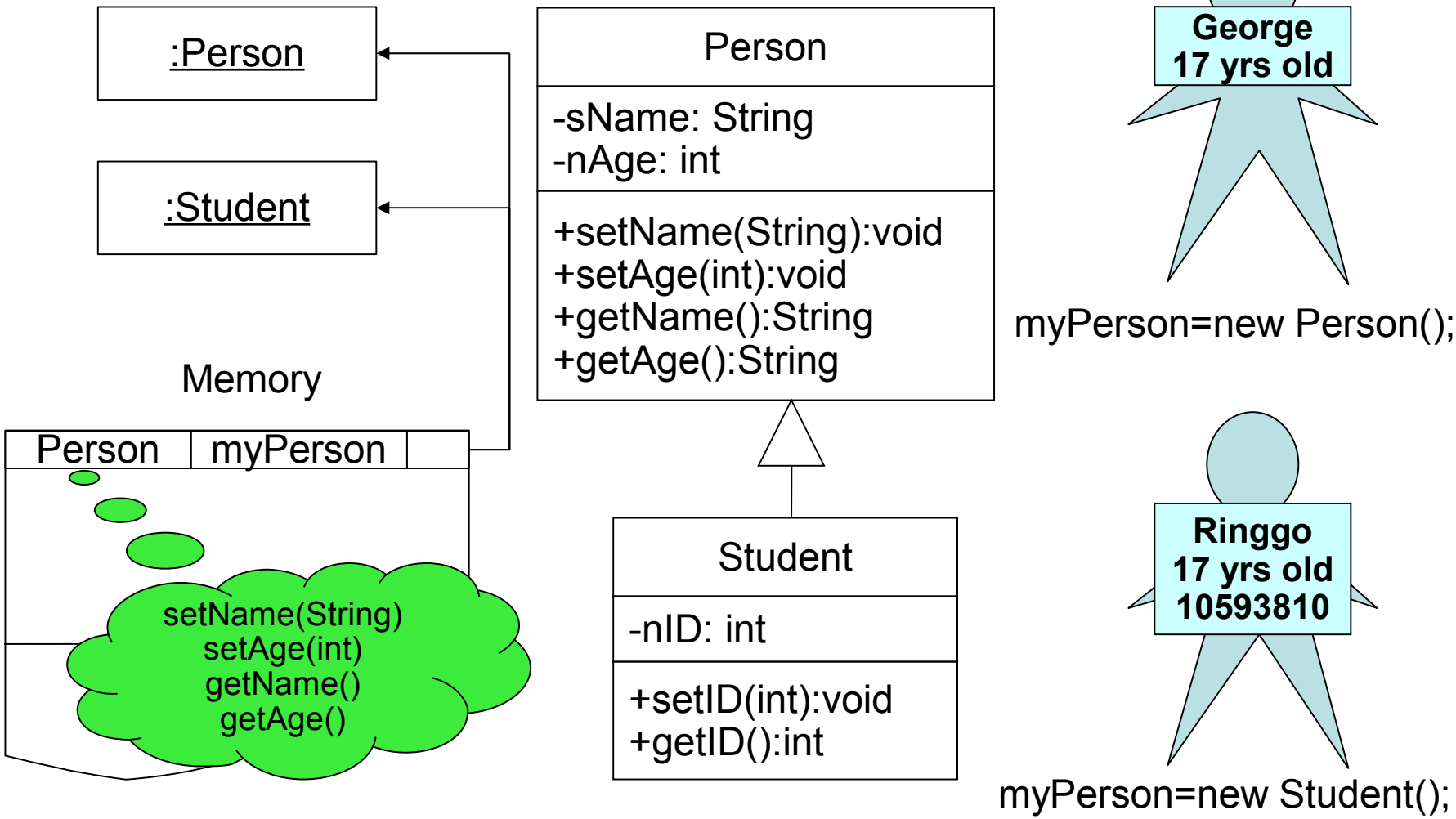
- What happens if there are very many different types of employees?

Inheritance in Java

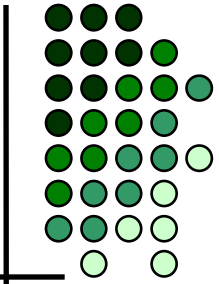


- Since a subclass extends the super class, then it is simply a more specialized type of the super class
- It is more specific than a superclass, but all the attributes and methods of a super class are still present in the subclass
- This allows an identifier assigned to the superclass type to point to a sub class object

Inheritance in Java



Inheritance in Java



```
public class Display
{
  public void displayEmployeeSalary(Employee eEmp)
  {
      System.out.println(“Name: ” + eEmp.getName());
      System.out.println("Salary: "+ eEmp.getSalary());
  }
}
public class Driver
{
   public static void main(String[] args)
  {
      Manager m=new Manager();
     Teller t=new Teller();
      Display d=new Display();
      d.displayEmployeeSalary(m);
      displayEmployeeSalary(t);
  }
}
```

Inheritance in Java

Paul Inventado

De La Salle University