JAVA Programming

Inheritance

Overview

- Inheritance
- Override
- Abstract
- Final
- Interfaces

Inheritance

- We have two kind of employees
  - Temporary Employees
  - Permanent employees

- Each employee has a name and id
- Each employee has a salary
  - For temporary employee, the salary is based on numberOfHours
  - For permanent employees, the salary is based on salaryPosition

Inheritance

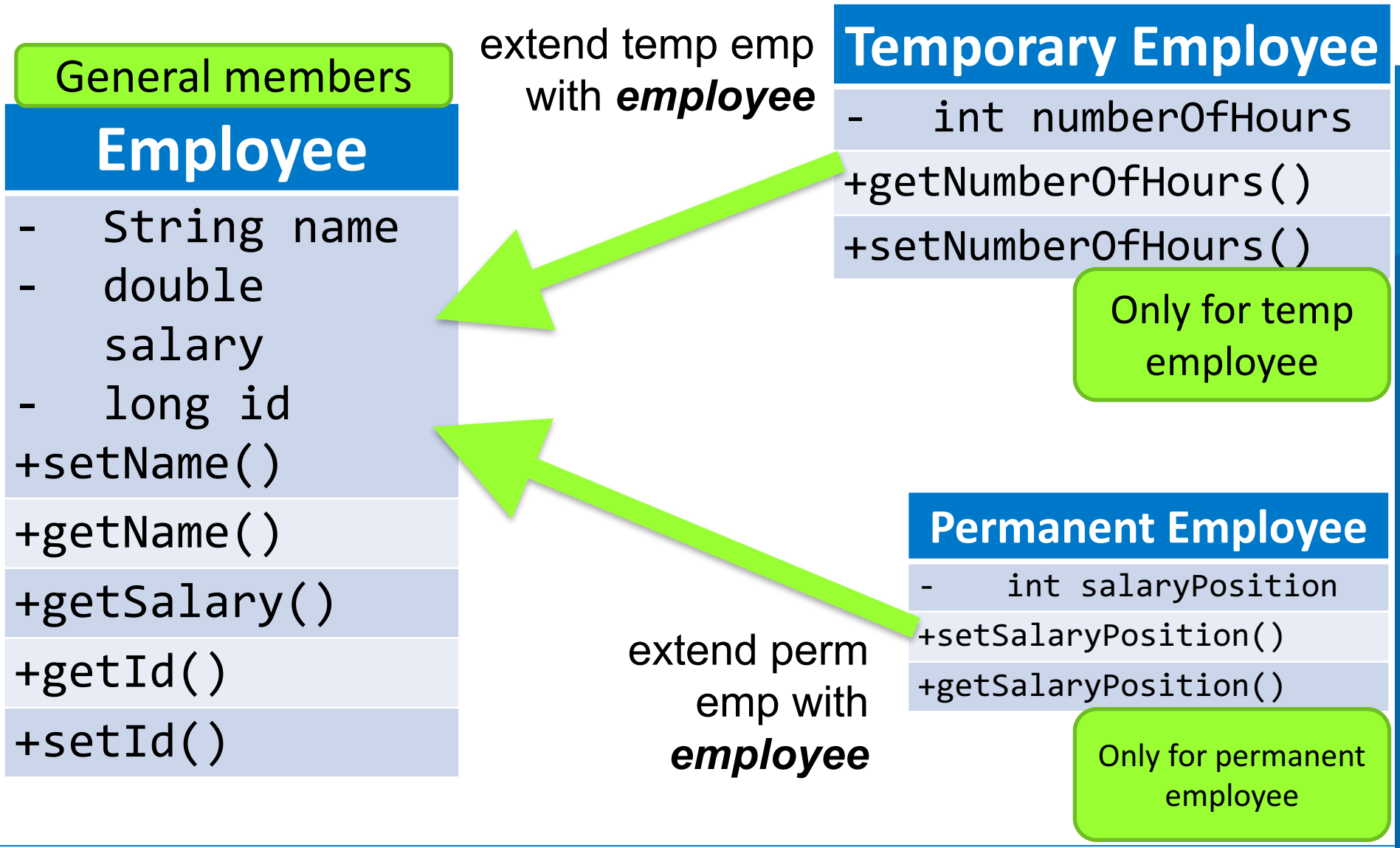
Permanent Employee
- String name
- double salary
- int salaryPosition
- long id
+setName()
+getName()
+getSalary()
+setSalaryPosition()
+getId()
+setId()

Temporary Employee
- String name
- double salary
- int numberOfHours
- long id
+setName()
+getName()
+getSalary()
+setNumberOfHours()
+getId()
+setId()

Both are employees…

We want to reuse parts of the code

Inheritance



Inheritance

- Create three classes:
  - Employee
  - PermanentEmployee
  - TemporaryEmployee
- PermanentEmployee and TemporaryEmployee extend the Employee Class
- Add specific members to PermanentEmployee and TemporaryEmployee

Inheritance

```
public class Employee {

 private long id;
 private String name;

 getters / setters;
}
```

Super class

Derived class

```
public class TemporaryEmployee extends Employee {

 private int numberOfHours;

 getters / setters;
}
```

Inheritance

- Make the Employee class abstract, as it will never be instantiated

```
public abstract class Employee {

 private long id;
 private String name;

 getters / setters;
}
```

Inheritance

- How do we implement the salary?

```
public class Employee {
 public long getSalary() {
    return -1;
 }
```

```
public class TemporaryEmployee
extends Employee{
 private int numberOfHours;
 public long getSalary() {
    return numberOfHours * 40;
 }
}
```

```
public class PermantentEmployee
extends Employee{
 private int position;
 public long getSalary() {
    return  2500 * position ;
 }
}
```

Inheritance

- What is the output of the following code snippet.

```
Employee employee;
PermanentEmployee permanentEmployee =
                                    new PermanentEmployee();
employee = permanentEmployee;
System.out.println(employee.getSalary);
```

Inheritance

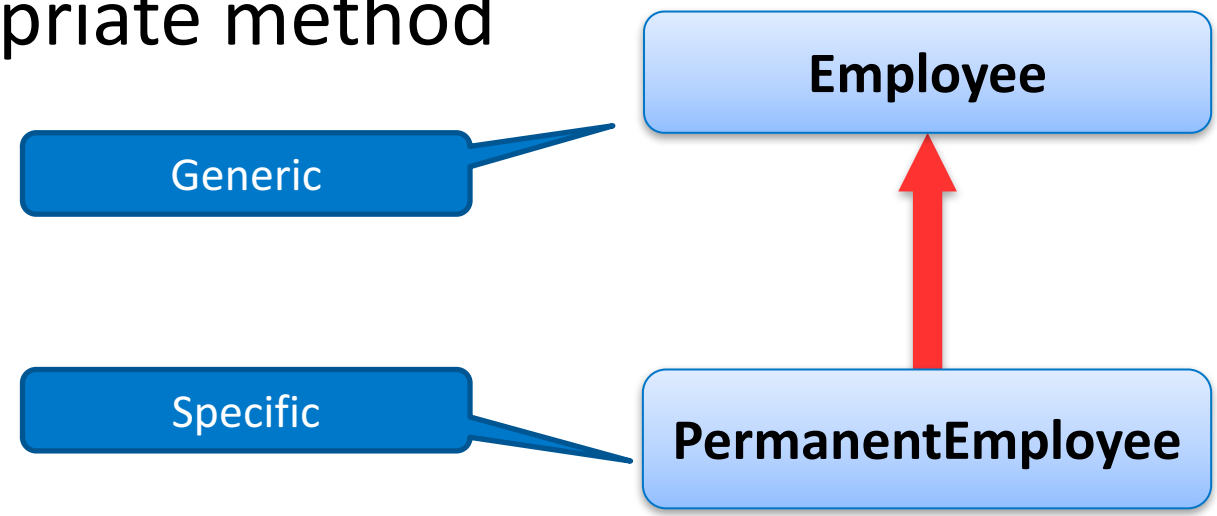
- What is the output of the following code snippet.

```
Employee employee;
PermanentEmployee permanentEmployee =
                                    new PermanentEmployee();
employee = permanentEmployee;
System.out.println(employee.getSalary);
```

- The most specific implementation of getSalary() is used.

Inheritance

- Output comes from the method in the most specific class
- PermanentEmployee **overrides** the getSalary() method of Employee
- Java begins at the most specific class to find an appropriate method



Inheritance

- **super** members are reachable

```
public class TemporaryEmployee {
 private int numberOfHours = 1;

 public String toString() {
   return getId() + " - " +
          getName();
 }

}
```

Inheritance

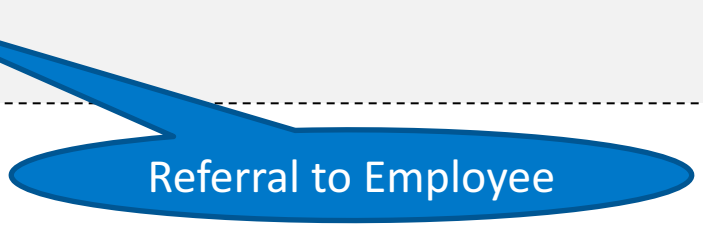
- Calling constructor of super class

```
public class Employee {

 public Employee(long id, String name) {
   this.setId(id);
   this.setName(name);
```

```
public class TemporaryEmployee extends Employee{

 public TemporaryEmployee(long id, String name, int hours) {
       super(id, name);
    setHours(hours);
```



Inheritance

- Calling methods from super class

```
public class TemporaryEmployee {

  public String toString() {
   return super.getId() + " - " +
          super.getName();
 }
```

Inheritance

- specialization / generalization
- only single inheritance
- Derived Class inherits all members, except:
  - constructors
  - finalizers
- implicit conversion from derived to super
- all classes implicitly derive from class Object
  - if no other base-class specified

Override

- Enables polymorphism
- Instance methods can be overridden by default
- implementation of methods can be overridden by derived class
  - runtime type of the instance determines which method is called
    - most derived methods
- Final methods are not overrideable

Override

- Instance method in subclass with same signature and return type as instance method in superclass overrides superclass' method
- Override method specializes the implementation of an inherited method
  - Super-method can be accessed using *super.*
  - No keyword needed
  - use @override to enable compiler checking

Hiding

- Static method in subclass with same signature and return type as static method in superclass hides superclass' method

Abstract

- abstract classes cannot be instantiated
  - intended to be a base class only
  - may contain abstract methods
    - methods without implementation
- Derived class usually implements all abstract methods, but if not, derived class must be abstract as well
- Abstract methods only in abstract classes

```
public abstract class Employee{
      public abstract void raiseSalary();
}
```

Final

- a final class cannot be subclassed
- prevents unintended derivation
- a final class cannot be abstract
- Many classes in java are final (like String, Integer and other wrapper classes)

- A final method can not be overridden.
- Object contains a number of final methods (getClass(), notify(), wait(), ..)

Interfaces

- Interfaces are like abstract classes, but
  - Only abstract methods, no implementation.
- Interface can declare
  - Constants (fields) implicitly static final
  - Methods implicitly abstract
  - Nested classes and interfaces
- All members are implicitly public

Interfaces

- interfaces can be derived from other interfaces
  - multiple inheritance
- Classes can implement one or more interfaces
- Standard java packages contain many interfaces (xxxable, like Cloneable, Runnable ..)

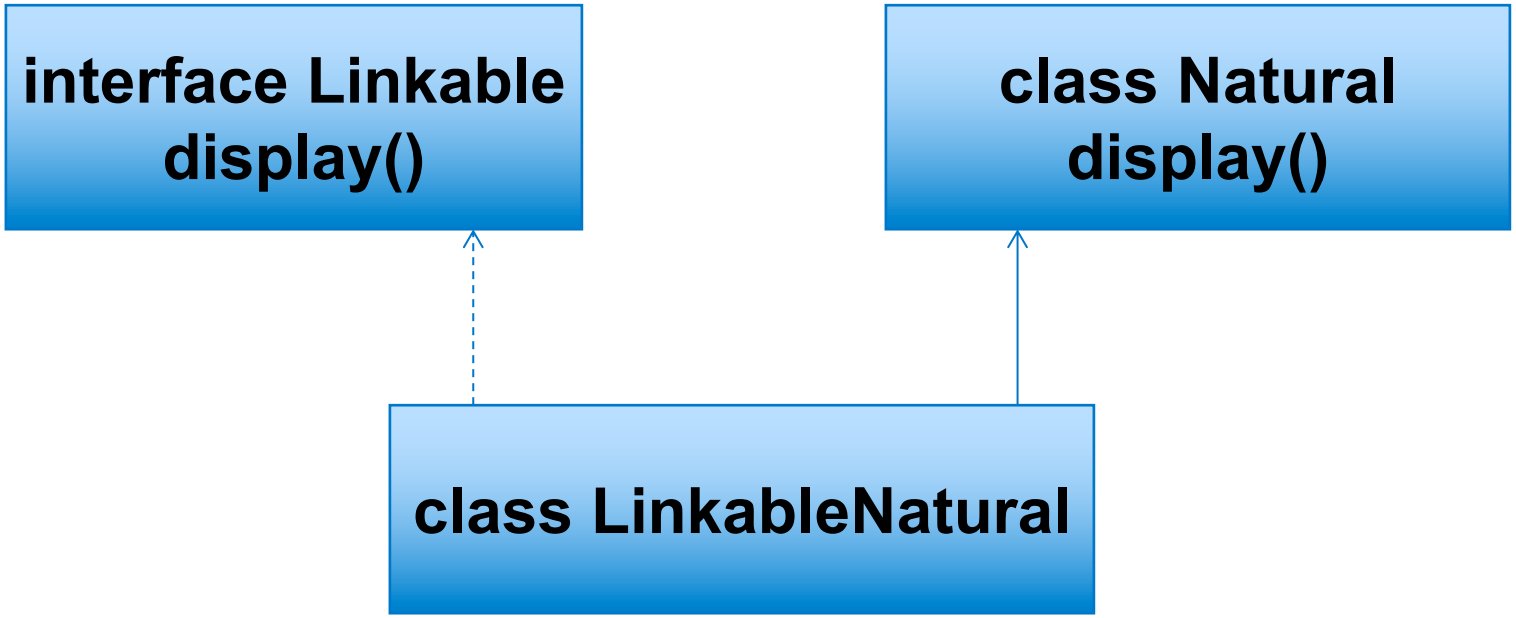
Interfaces

```
public interface Comparable {
int compareTo(Object o);
}
```

```
public class Employee implements Comparable {
      public int compareTo(Object o) {
        Employee temp = (Employee) o;
        if (getEmpID() == temp.getEmpID()) {
            return 0;
        } else if (getEmpID() < temp.getEmpID()) {
            return -1;
        } else {
            return 1;
      }
}
```

interface implementation

- interface-methods can be provided by base class!



Lab: Inheritance and interfaces