CSE 1201
Object Oriented Programming

Polymorphism

Acknowledgement

- For preparing the slides I took materials from the following sources
  - Course Slides of Dr. Tagrul Dayar, Bilkent University
  - Java book "*Java Software Solutions*" by Lewis & Loftus.

Polymorphism

- The term *polymorphism* literally means "having many forms"
- A *polymorphic reference* is a variable that can refer to different types of objects at different points in time
- The method invoked through a polymorphic reference can change from one invocation to the next
- All object references in Java are potentially polymorphic

Polymorphism

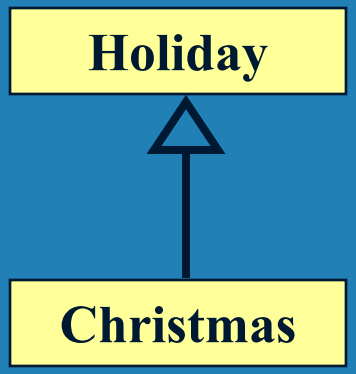
- Suppose we create the following reference variable:

```
Occupation job;
```

- Java allows this reference to point to an `Occupation` object, or to any object of any compatible type
- This compatibility can be established using inheritance or using interfaces
- Careful use of polymorphic references can lead to elegant, robust software designs

References and Inheritance

- An object reference can refer to an object of its class, or to an object of any class related to it by inheritance
- For example, if the `Holiday` class is used to derive a child class called `Christmas`, then a `Holiday` reference could be used to point to a `Christmas` object

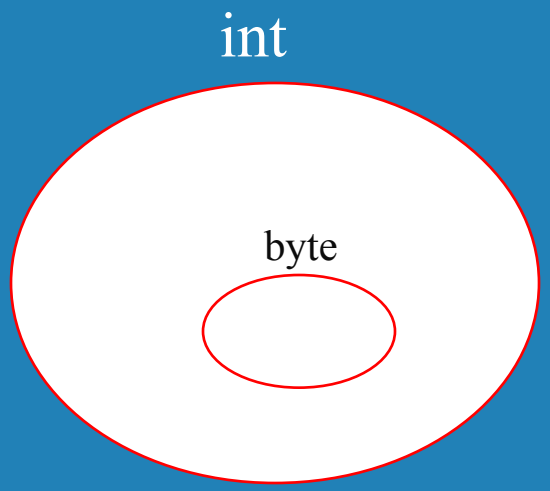


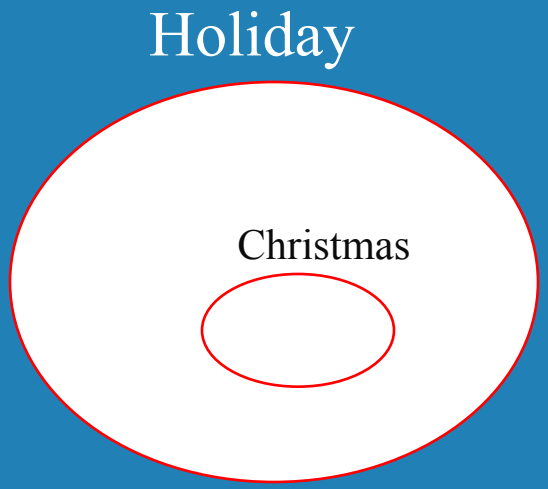

```
Holiday day;
day = new Christmas();
```

References and Inheritance

- Assigning a predecessor object to an ancestor reference is considered to be a widening conversion, and can be performed by simple assignment
- Assigning an ancestor object to a predecessor reference can be done also, but it is considered to be a narrowing conversion and must be done with a cast
- The widening conversion is the most useful
- An `Object` reference can be used to refer to any object
  - An `ArrayList` is designed to hold `Object` references

References and Inheritance





The set of int values is a wider set than the set of byte values, and contains all members of the byte values set.

```
byte b =  2;

int a = b; //widening conversion

b = a; // narrowing conversion, invalid

b = (byte) a; // this is ok

Holiday h = new Holiday(…);

Christmas ch = h; // invalid, not all
                  //holidays are christmas
```

Polymorphism via Inheritance

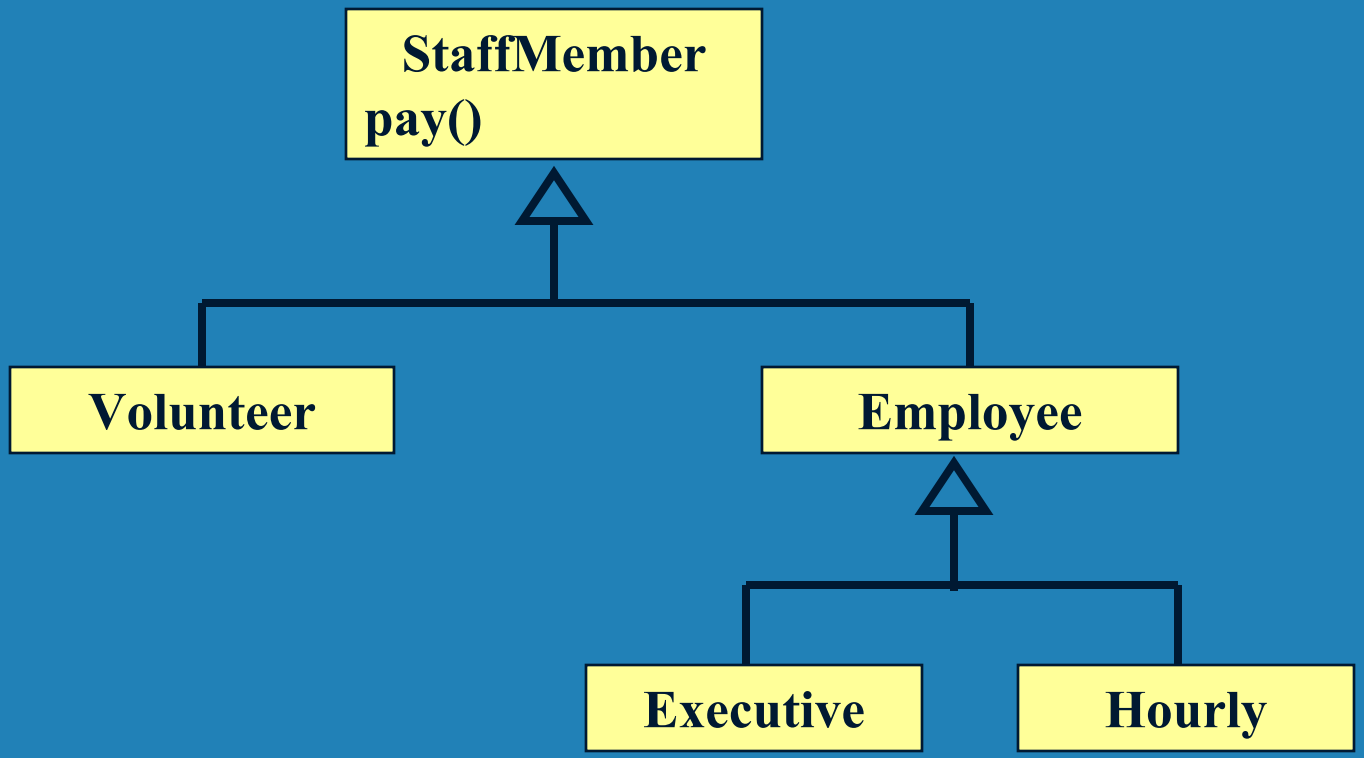
- It is the type of the object being referenced, not the reference type, that determines which method is invoked
- Suppose the `Holiday` class has a method called `celebrate`, and the `Christmas` class overrides it
- Now consider the following invocation:

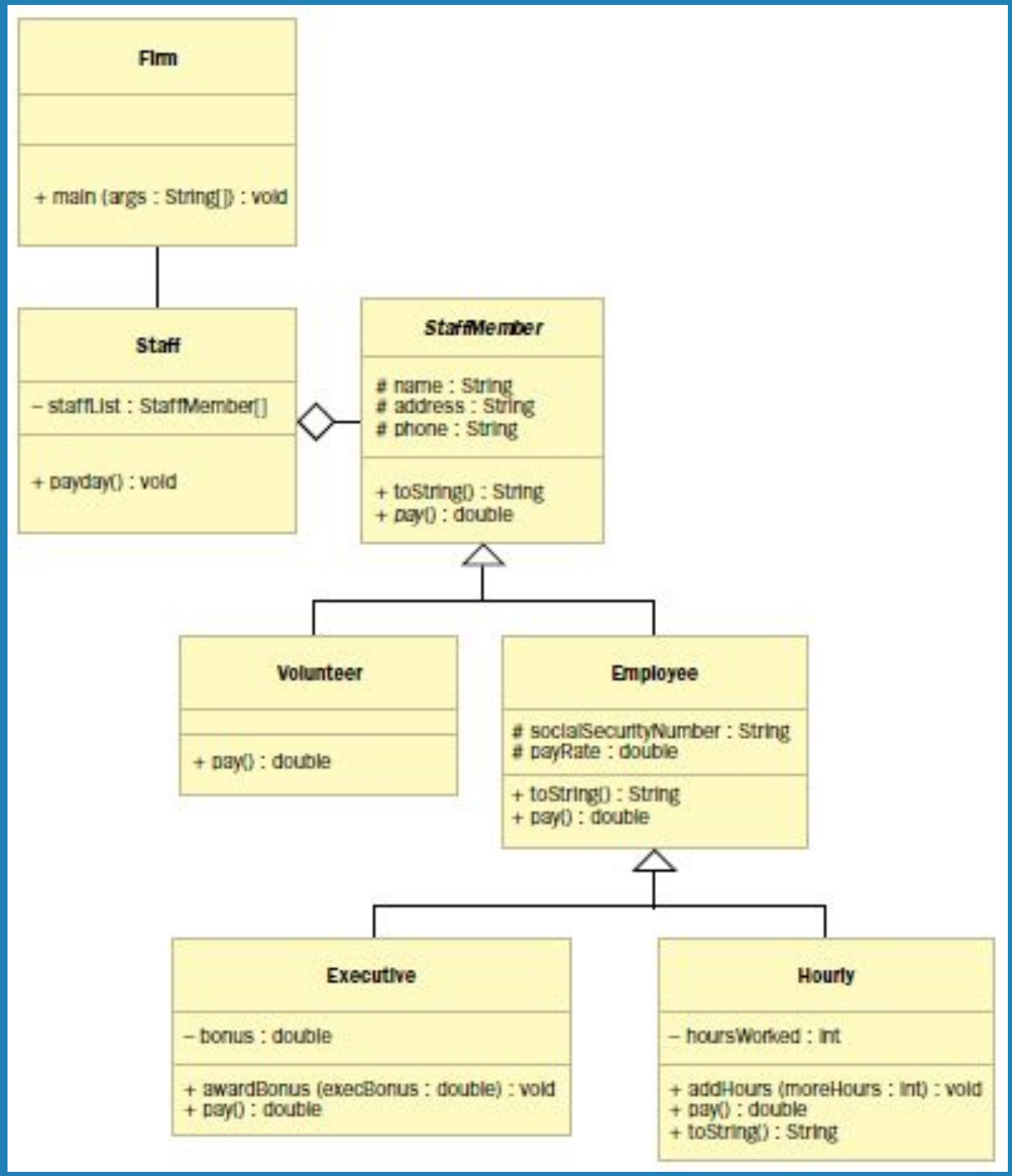
```
day.celebrate();
```

- If `day` refers to a `Holiday` object, it invokes the `Holiday` version of `celebrate`; if it refers to a `Christmas` object, it invokes the `Christmas` version

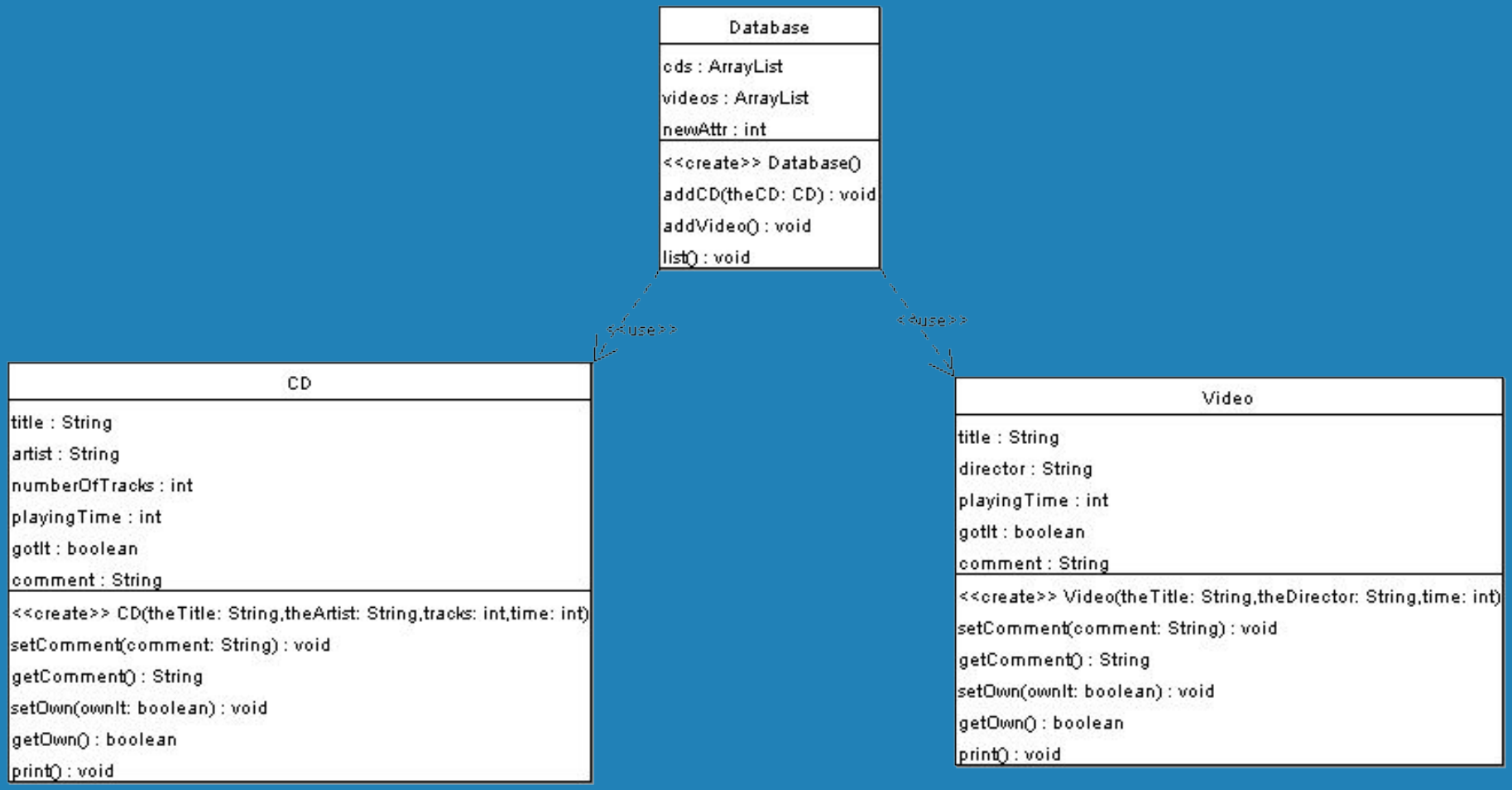
Polymorphism via Inheritance

- Consider the following class hierarchy:




Firm
+ main (args : String[]) : void
Staff
– staffList : StaffMember[]
+ payday() : void
StaffMember
name : String
address : String
phone : String
+ toString() : String
+ pay() : double
Volunteer
+ pay() : double
Employee
socialSecurityNumber : String
payRate : double
+ toString() : String
+ pay() : double
Executive
– bonus : double
+ awardBonus (execBonus : double) : void
+ pay() : double
Hourly
– hoursWorked : int
+ addHours (moreHours : int) : void
+ pay() : double
+ toString() : String

CD and Video Database



CD and Video Database revisited