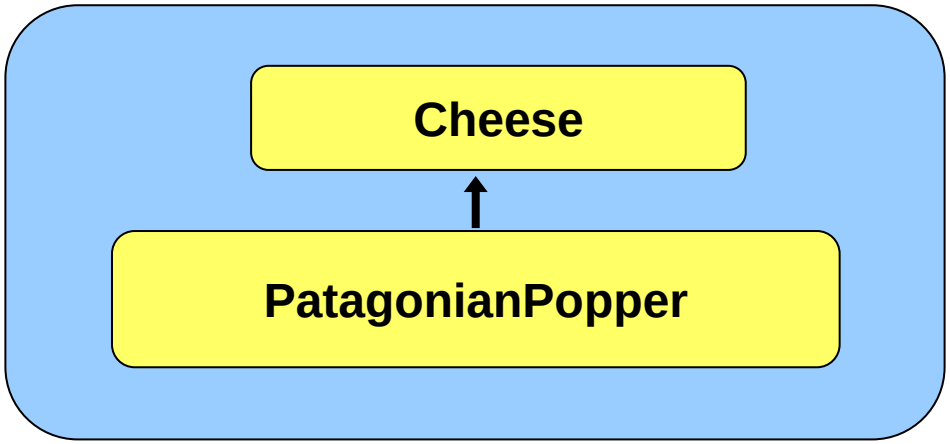
Class Hierarchies and Inheritance

Class Hierarchies

Inheritance

Access Modifiers

Class Hierarchies



http://computationaltales.blogspot.ca/2011/09/inheritance-in-cheeses-and-magic-spells.html

Cheese

milk:

weight:

density:

Cheese Methods

```
void emitSmell()
{ … }
```

PatagonianPopper

milk:

weight:

density:

poppiness:

PatagonianPopper Methods

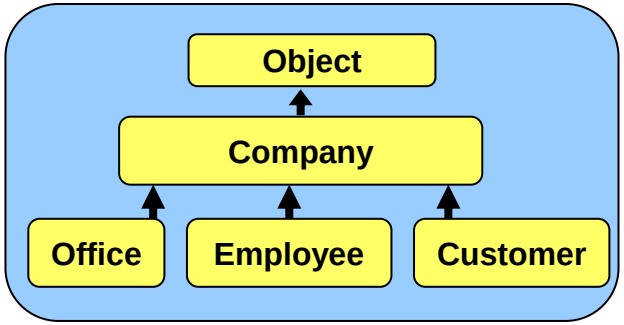
```
void poppingSensation()
{ … }
```

http://computationaltales.blogspot.ca/2011/09/inheritance-in-cheeses-and-magic-spells.html

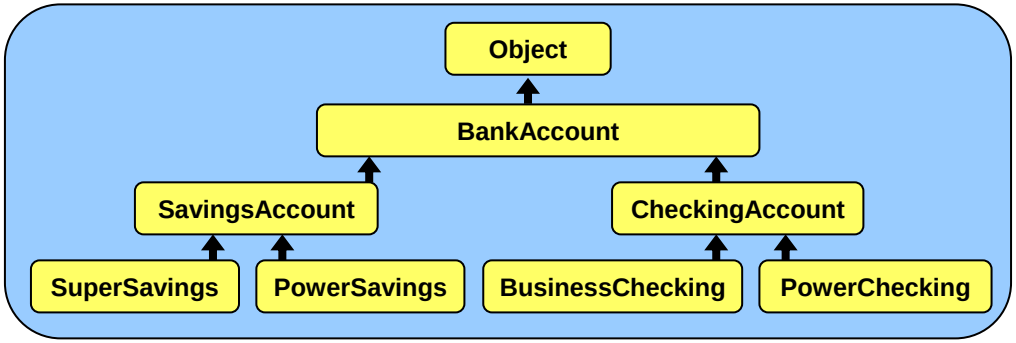
Poll Everywhere Question

Which of the following is an example of a good class hierarchy? **Text 37607**

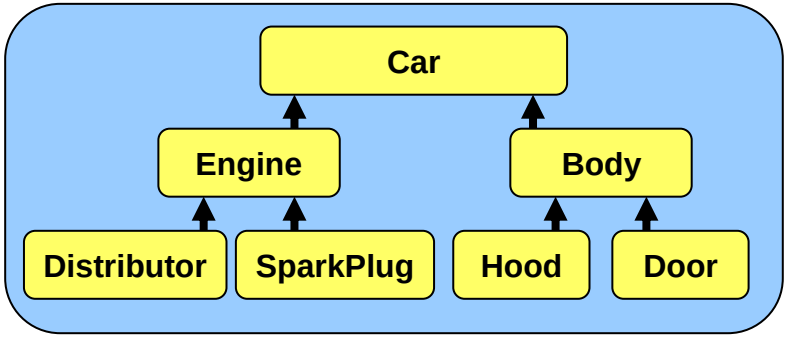
A: 201642



B: 201645

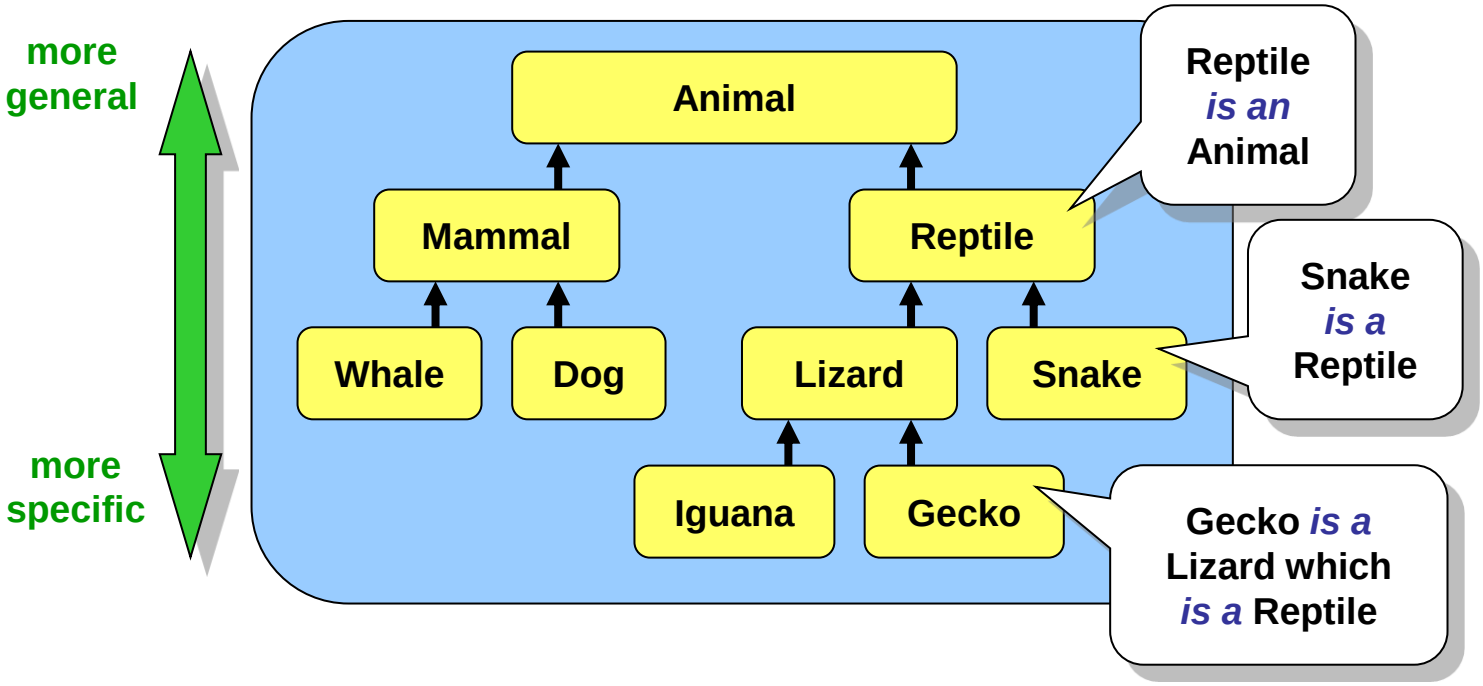


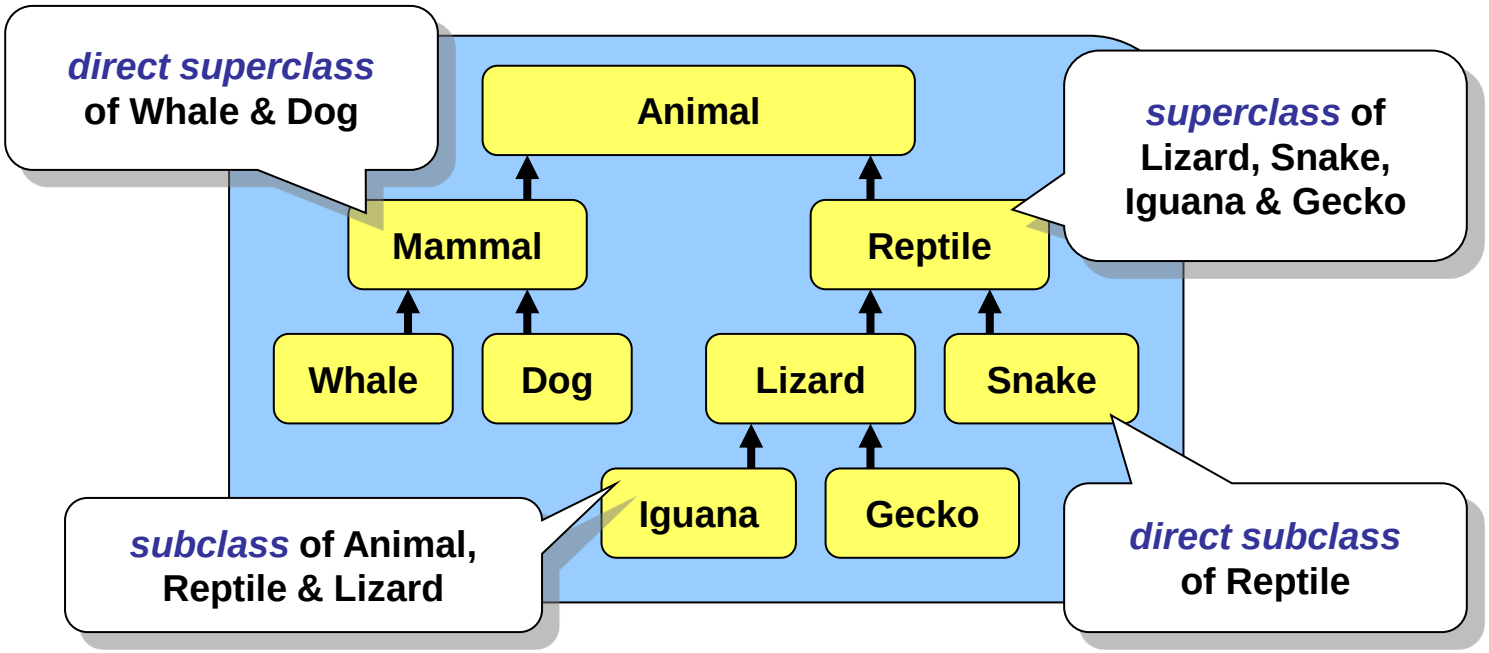
C: 201653

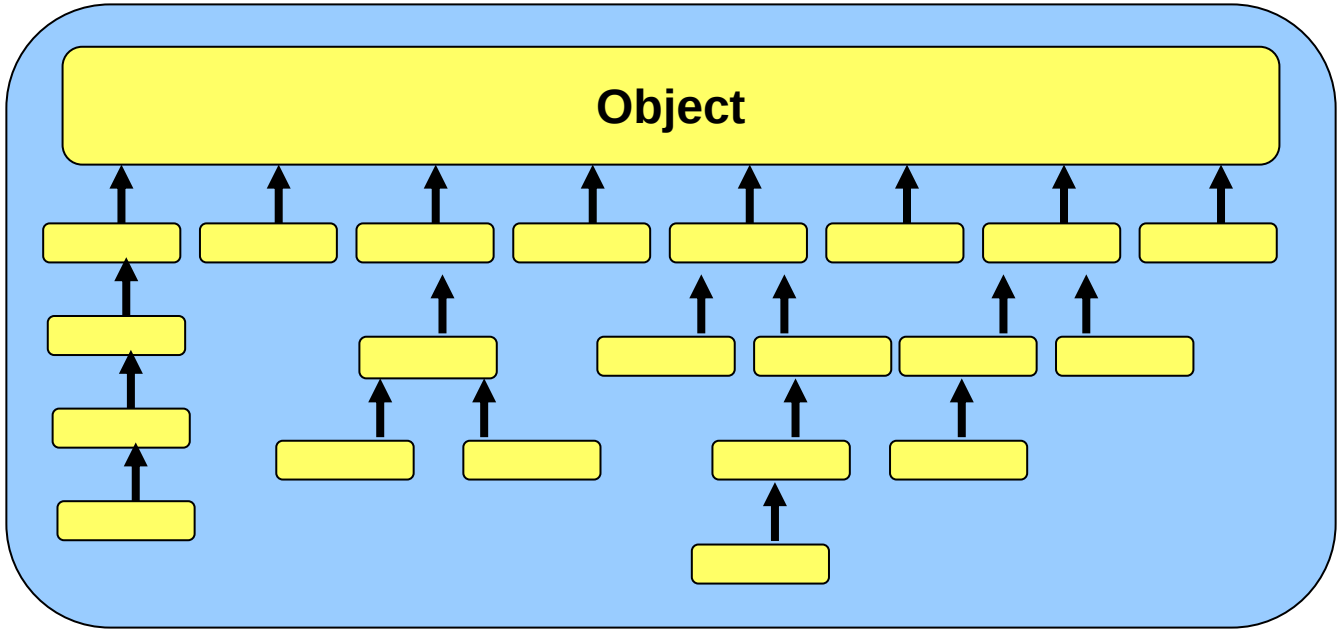


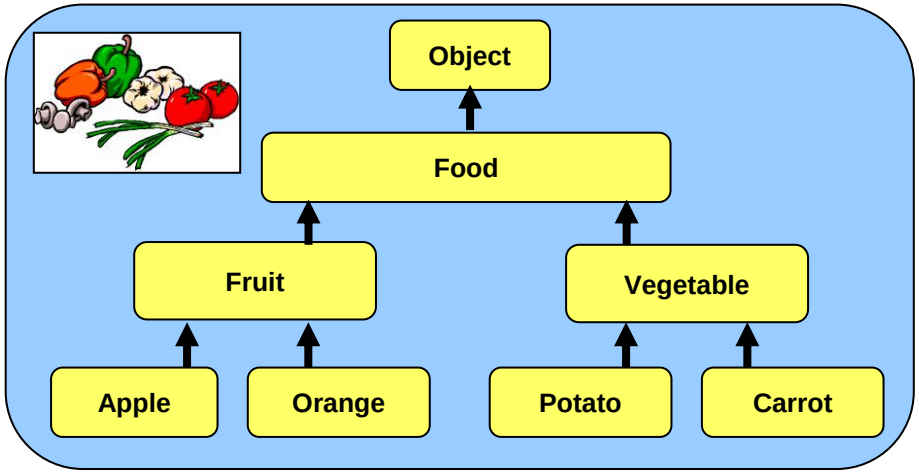
Class Hierarchy

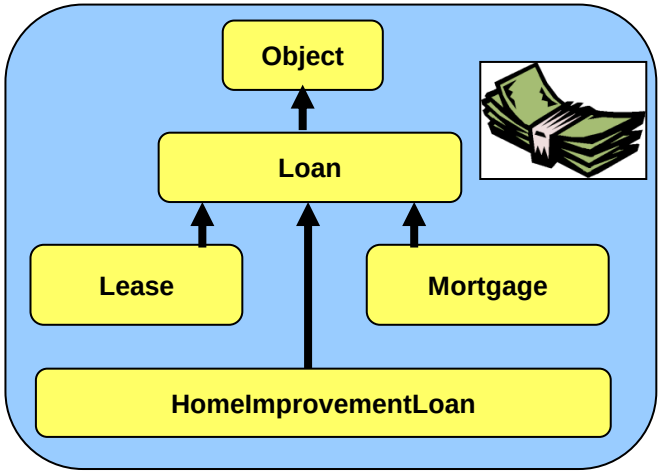
“is a” relationship

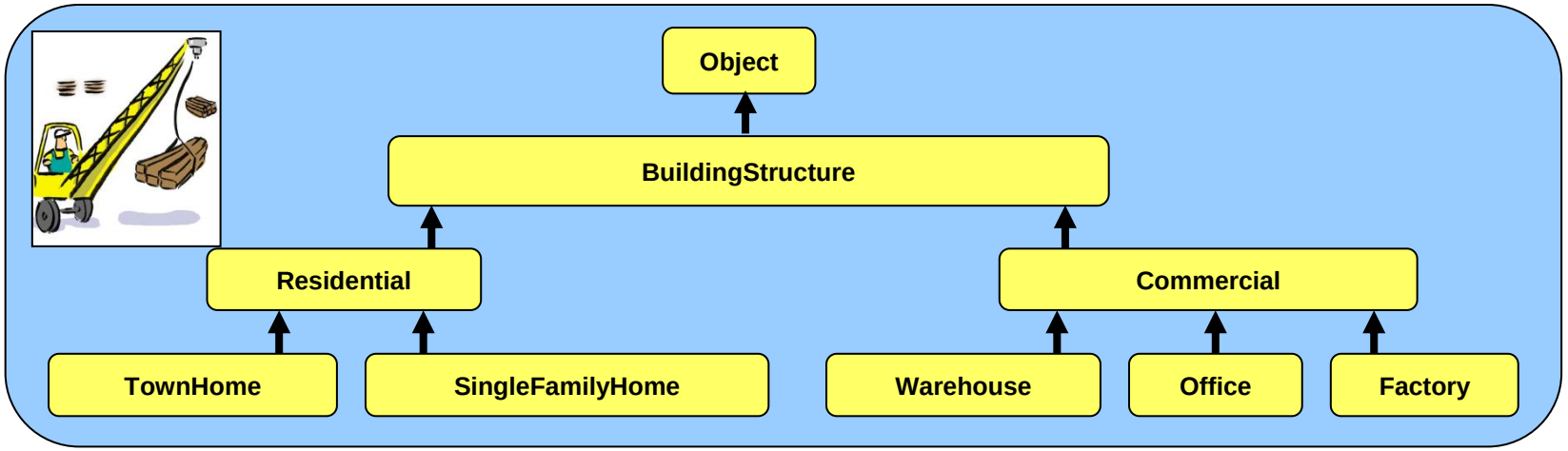

more general
more specific
Animal
Mammal
Reptile
Whale
Dog
Lizard
Snake
Iguana
Gecko
Reptile is an Animal
Snake is a Reptile
Gecko is a Lizard which is a Reptile


direct superclass
of Whale & Dog
Animal
superclass of
Lizard, Snake,
Iguana & Gecko
Mammal
Reptile
Whale
Dog
Lizard
Snake
Iguana
Gecko
subclass of Animal,
Reptile & Lizard
direct subclass
of Reptile


Object


Object
Food
Fruit
Vegetable
Apple
Orange
Potato
Carrot


Object
Loan
Lease
Mortgage
HomeImprovementLoan

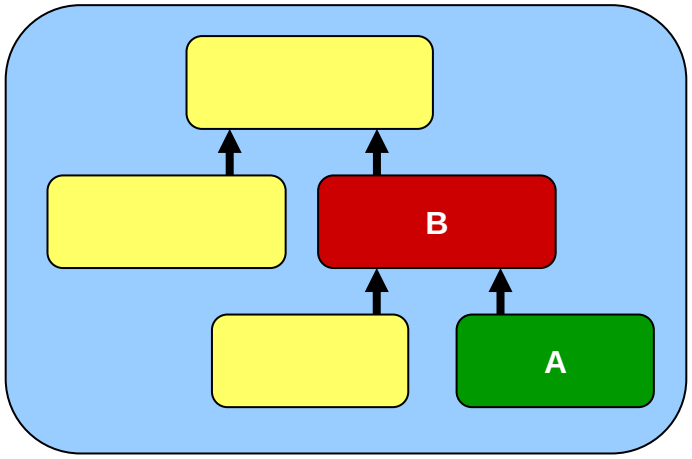

Object
BuildingStructure
Residential
Commercial
TownHome
SingleFamilyHome
Warehouse
Office
Factory

Class Hierarchy

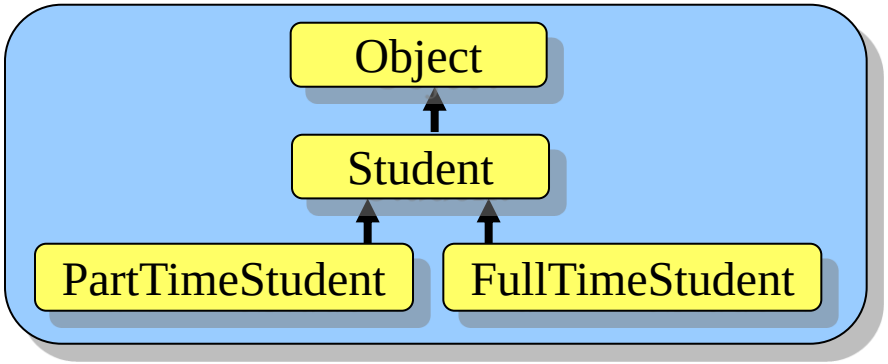
“has a” relationship?

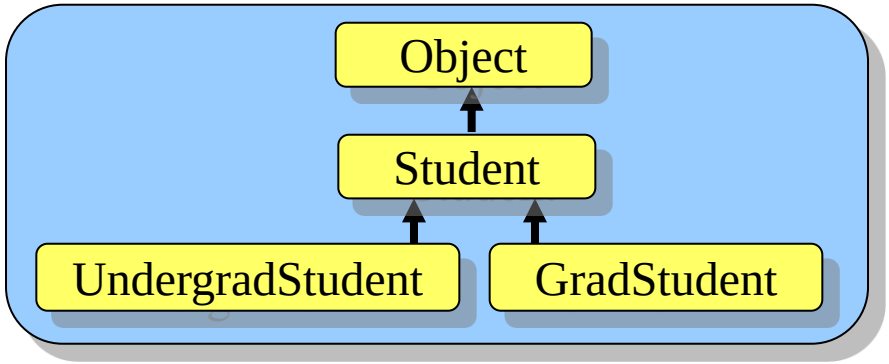

Class Hierarchy
“has a” relationship

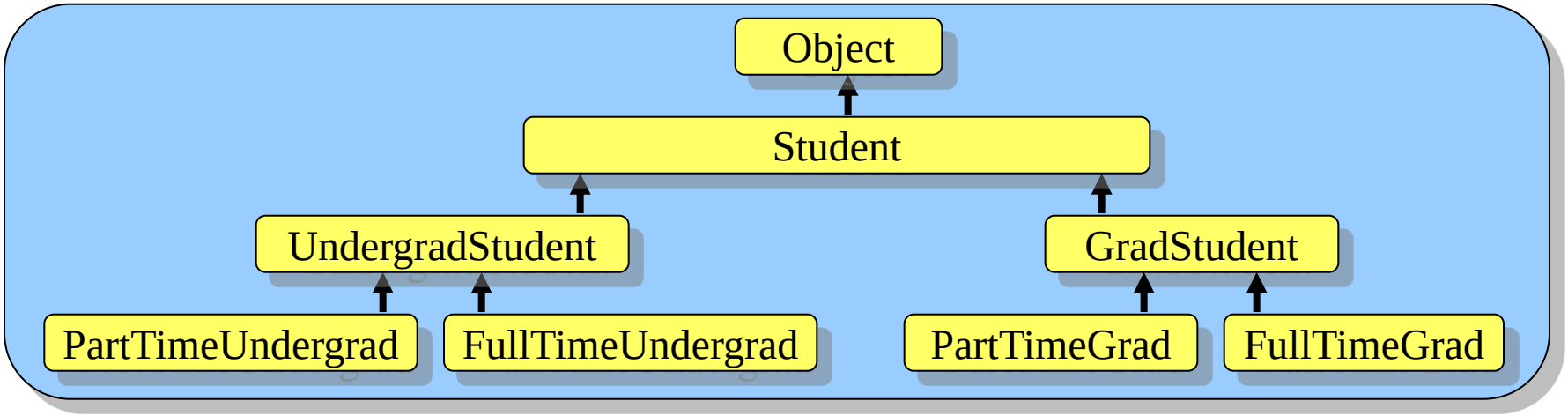
```
public class A extends B
{
     ...
}
```

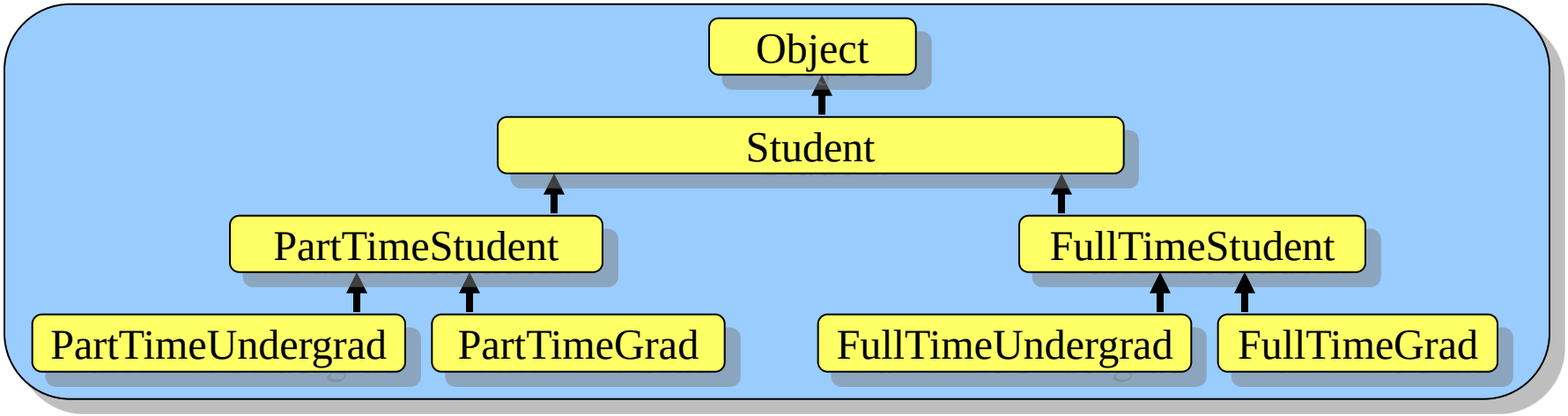


How do we know how to arrange our hierarchies?


Object
Student
PartTimeStudent
FullTimeStudent

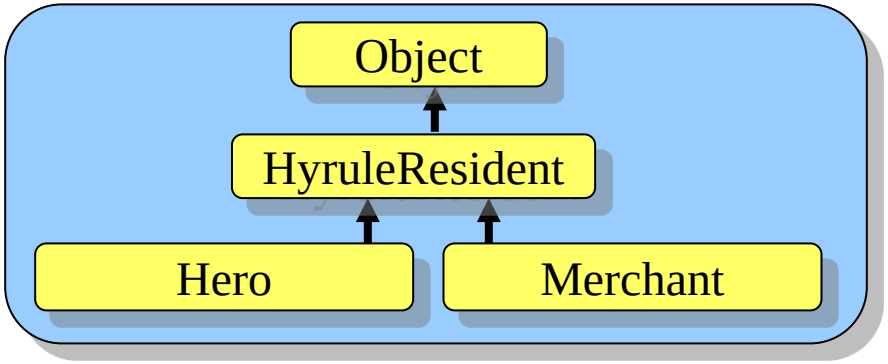

Object
Student
UndergradStudent
GradStudent

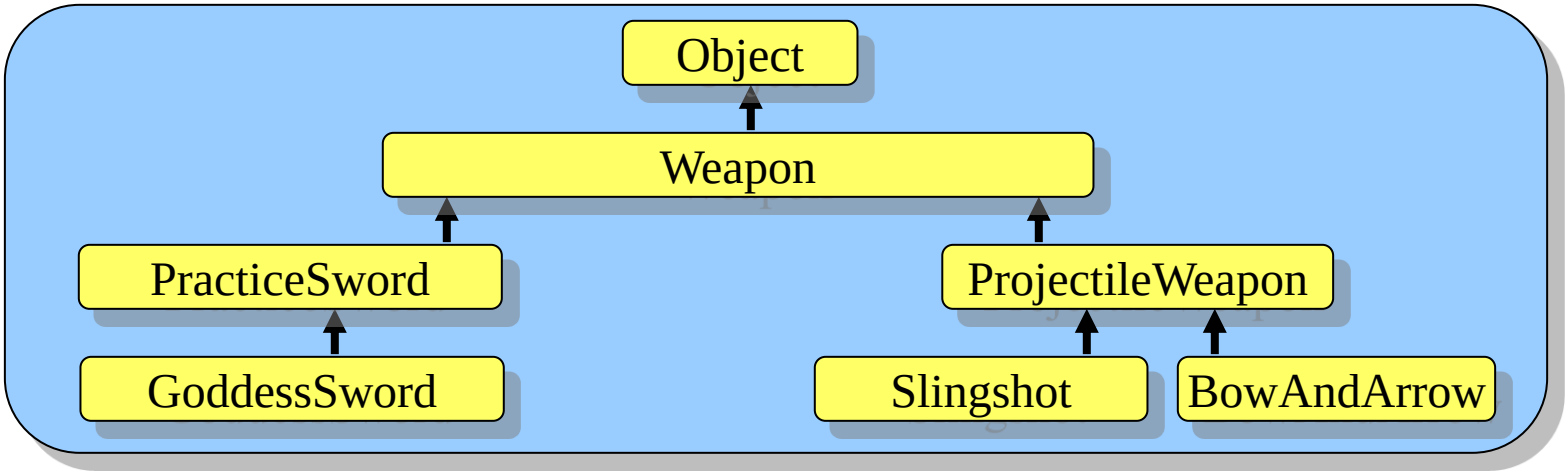

Object
Student
UndergradStudent
GradStudent
PartTimeUndergrad
FullTimeUndergrad
PartTimeGrad
FullTimeGrad


Object
Student
PartTimeStudent
FullTimeStudent
PartTimeUndergrad
PartTimeGrad
FullTimeUndergrad
FullTimeGrad

Answer: It depends!

Sometimes you change your mind later.


Object
HyruleResident
Hero
Merchant


Object
Weapon
PracticeSword
ProjectileWeapon
GoddessSword
Slingshot
BowAndArrow

Inheritance

Inheritance: the act of receiving shared attributes and behavior from more general types of objects up the hierarchy

Why Inheritance?

More **reusable** (code shared between classes)

Faster to program (less code repetition)

Simpler code (closer to real life)

```
public class Employee
{
        String name;
        Address address;
        String phoneNumber;
        int employeeNumber;
        float hourlyPay;
        …
}
```

```
public class Employee
{
        String name;
        Address address;
        String phoneNumber;
        int employeeNumber;
        float hourlyPay;
        …
}
```

Manager: Same as employee but with higher pay

```
public class Employee
{
        String name;
        Address address;
        String phoneNumber;
        int employeeNumber;
        float hourlyPay;
        …
}
```

Manager: Same as employee but with higher pay

no new class needed

```
public class Employee
{
        String name;
        Address address;
        String phoneNumber;
        int employeeNumber;
        float hourlyPay;
        …
}
```

Manager: Same as employee with additional attributes

```
public class Employee
{
        String name;
        Address address;
        String phoneNumber;
        int employeeNumber;
        float hourlyPay;
        …
}
```

Manager: Same as employee with additional attributes

create a subclass

```
public class Employee
{
        String name;
        Address address;
        String phoneNumber;
        int employeeNumber;
        float hourlyPay;
        …
}

public class Manager extends Employee
{
        String[] duties;
        Employee[] subordinates;
        …
}
```

```
public class Employee
{
        String name;
        Address address;
        String phoneNumber;
        int employeeNumber;
        float hourlyPay;
        …
}

public class Manager extends Employee
{
        String[] duties;
        Employee[] subordinates;
        …
}
```

5 attributes

```
public class Employee
{
        String name;
        Address address;
        String phoneNumber;
        int employeeNumber;
        float hourlyPay;
        …
}

public class Manager extends Employee
{
        String[] duties;
        Employee[] subordinates;
        …
}
```

5 `Employee` attributes plus 2 more – total 7 attributes

Employee

name:

address:

phoneNumber:

employeeNumber:

hourlyPay:

Manager

name:

address:

phoneNumber:

employeeNumber:

hourlyPay:

duties:

subordinates:

Employee Methods

...

Manager Methods

...

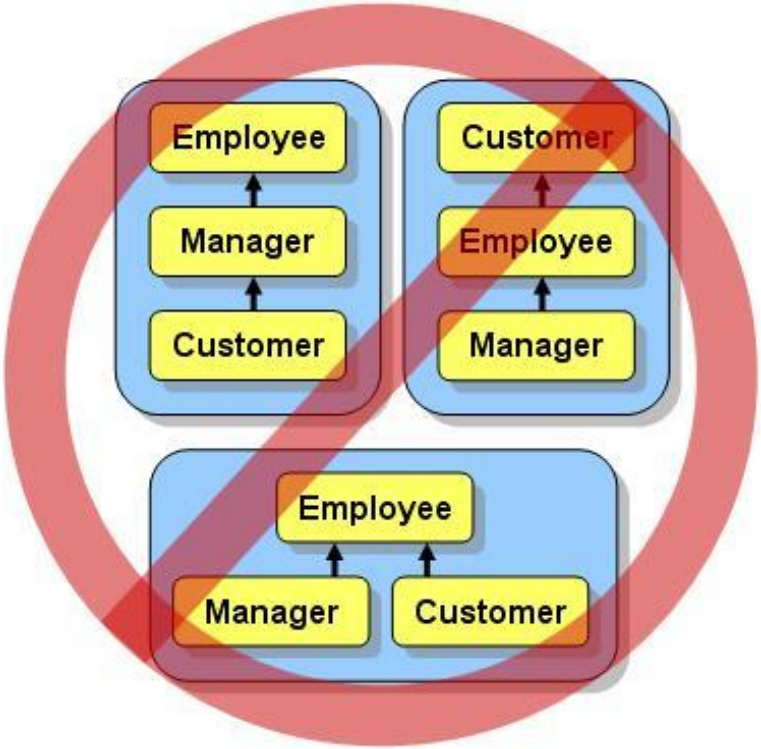
What if we wanted to add a customer?

```
public class Customer
{
        String name;
        Address address;
        String phoneNumber;
        …
}
```

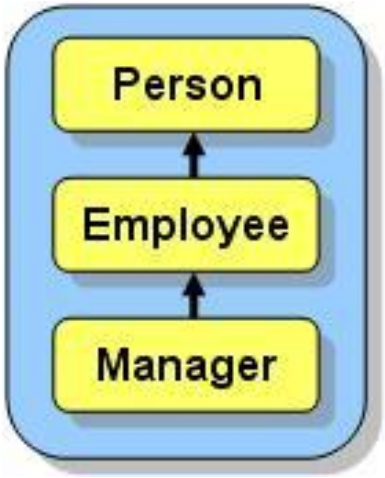
```
public class Customer
{
    String name;
    Address address;
    String phoneNumber;
    …
}
```

Some common attributes with Employee

```
public class Customer
{
        String name;
        Address address;
        String phoneNumber;
        …
}
```

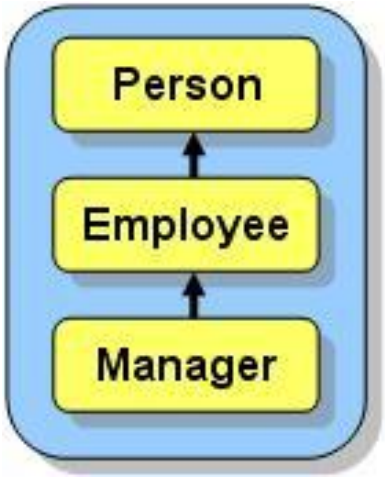


```
public class Person
{
      String       name;
      Address      address;
      String       phoneNumber;
}
public class Employee extends Person
{
      int          employeeNumber;
      float        hourlyPay;
}
public class Manager extends Employee
{
      String[]       duties;
      Employee[]     subordinates;
}
```



```
public class Person
{
      String       name;
      Address      address;
      String       phoneNumber;
}
public class
{
      int
      float        hourlyPay;
}
public class Manager extends Employee
{
      String[]       duties;
      Employee[]     subordinates;
}
```

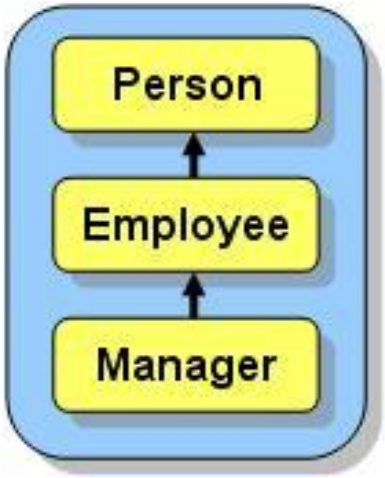
Represent customers with `Person`



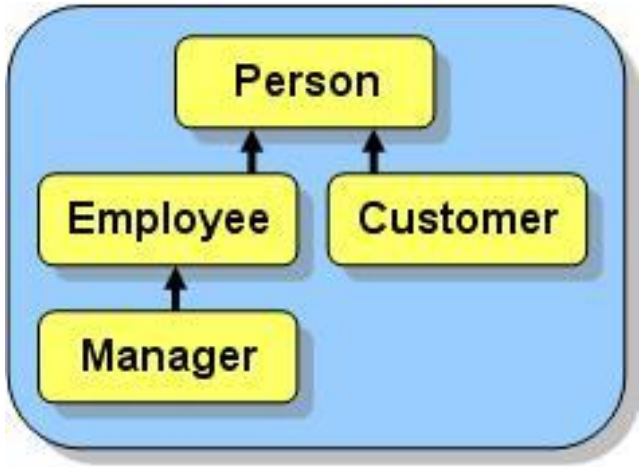
```
public class Person
{
        String          name;
        Address         address;
        String          phoneNumber;
}
public class
{
        int
        float
}
public class
{
        String[]
        Employee[]      subordinates;
}
```

Represent customers with `Person`

All attributes of a `Person` must be relevant to subclasses

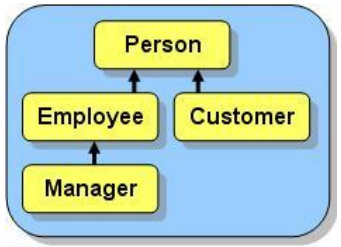


```
public class Person
{
        String  name;
        Address address;
        String  phoneNumber;
}

public class Employee extends Person
{
        int employeeNumber;
        float hourlyPay;
}

public class Customer extends Person
{
        String[] itemsPurchased;
        Date[] purchaseHistory;
}

public class Manager extends Employee
{
        String[] duties;
        Employee[] subordinates;
}
```



If needed, we can make a `Customer` subclass of Person

```
Person       p = new Person();
Employee     e = new Employee();
Customer     c = new Customer();
Manager      m = new Manager();

p.name = "Hank Urchiff";
p.address = new Address();
p.phoneNumber = "1-613-555-2328";

e.name = "Minnie Mumwage";
e.address = new Address();
e.phoneNumber = "1-613-555-1231";
e.employeeNumber = 232867;
e.hourlyPay = 8.75f;
```

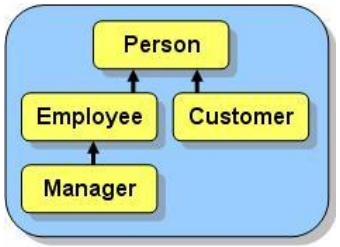


```
Person      p = new Person();
Employee    e = new Employee();
Customer
Manager
```

Own attributes

```
p.name = "Hank Urchiff";
p.address = new Address();
p.phoneNumber = "1-613-555-2328";
```

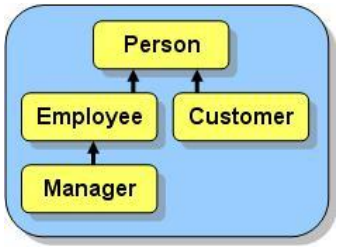
```
e.name = "Minnie Mumwage";
e.address = new Address();
e.phoneNumber = "1-613-555-1231";
e.employeeNumber = 232867;
e.hourlyPay = 8.75f;
```



```
Person       p = new Person();
Employee     e = new Employee();
Customer     c = new Customer();
Manager      m = new Manager();

p.name = "Hanl
p.address = ne
p.phoneNumber
```

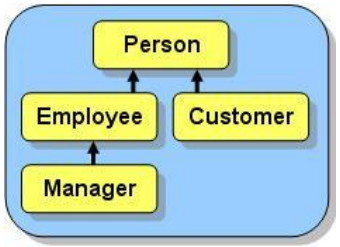
Attributes inherited from `Person`

```
e.name = "Minnie Mumwage";
e.address = new Address();
e.phoneNumber = "1-613-555-1231";
e.employeeNumber = 232867;
e.hourlyPay = 8.75f;
```

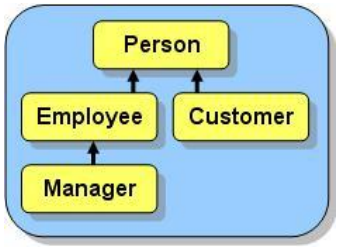


```
Person       p = new Person();
Employee     e = new Employee();
Customer     c = new Customer();
Manager      m = new Manager();

p.name = "Hank Urchiff";
p.address = new Address();
p.phoneNumber = "1-613-555-2328";

e.name = "Minnie Mumwage";
e.address = new Address();
e.phoneNumber = "1-613-555-1231";
e.employeeNumber = 232867;
e.hourlyPay = 8.75f;
```

Own attributes

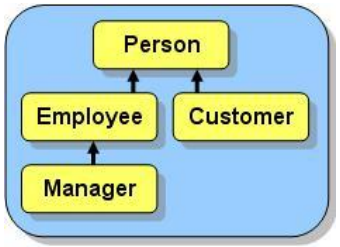


```
c.name = "Jim Clothes";
c.address = new Address();
c.phoneNumber = "1-613-555-5675";
c.itemsPurchased[0] = "Pencil Case";
c.purchaseHistory[0] = Date.today();

m.name = "Max E. Mumwage";
m.address = new Address();
m.phoneNumber = "1-613-555-8732";
m.employeeNumber = 232867;
m.hourlyPay = 8.75f;
m.duties[0] = "Phone Clients";
m.subordinates[0] = e;
```



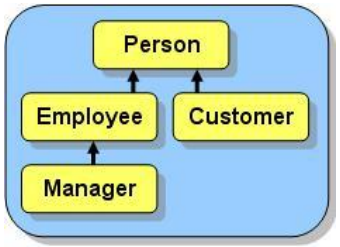
Attributes inherited from `Person`

```
c.name = "Jim Clothes";
c.address = new Address();
c.phoneNumber = "1-613-555-5675";
c.itemsPurchased[0] = "Pencil Case";
c.purchaseHistory[0] = Date.today();

m.name = "Max E. Mumwage";
m.address = new Address();
m.phoneNumber = "1-613-555-8732";
m.employeeNumber = 232867;
m.hourlyPay = 8.75f;
m.duties[0] = "Phone Clients";
m.subordinates[0] = e;
```



```
c.name = "Jim Clothes";
c.address = new Address();
c.phoneNumber = "1-613-555-5675";
c.itemsPurchased[0] = "Pencil Case";
c.purchaseHistory[0] = Date.today();

m.name = "M
m.address = new Address();
m.phoneNumber = "1-613-555-8732";
m.employeeNumber = 232867;
m.hourlyPay = 8.75f;
m.duties[0] = "Phone Clients";
m.subordinates[0] = e;
```

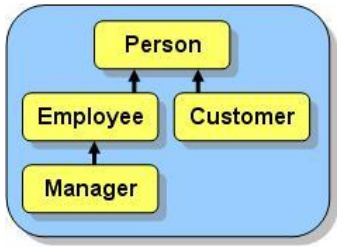
Own attributes



```
c.name = "Jim Clothes";
c.address = new Address();
c.phoneNum
c.itemsPur                                    e";
c.purchase                                    ();
```

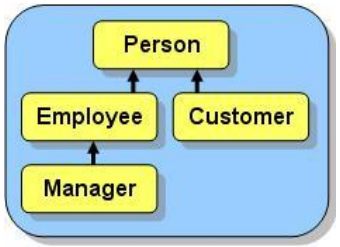
Attributes inherited from `Person`

```
m.name = "Max E. Mumwage";
m.address = new Address();
m.phoneNumber = "1-613-555-8732";
m.employeeNumber = 232867;
m.hourlyPay = 8.75f;
m.duties[0] = "Phone Clients";
m.subordinates[0] = e;
```



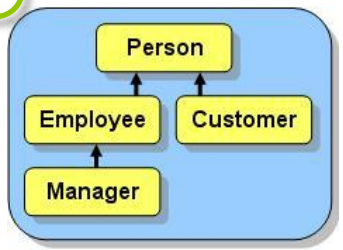
```
c.name = "Jim Clothes";
c.address = new Address();
c.phoneNumber = "1-613-555-5675";
c.itemsPurchased[0] = "Pencil Case";
c.purchaseHistory[0] = Date.today();

m.name = "
m.address
m.phoneNumber = 1-613-555-8732;
m.employeeNumber = 232867;
m.hourlyPay = 8.75f;
m.duties[0] = "Phone Clients";
m.subordinates[0] = e;
```

Attributes inherited from `Employee`

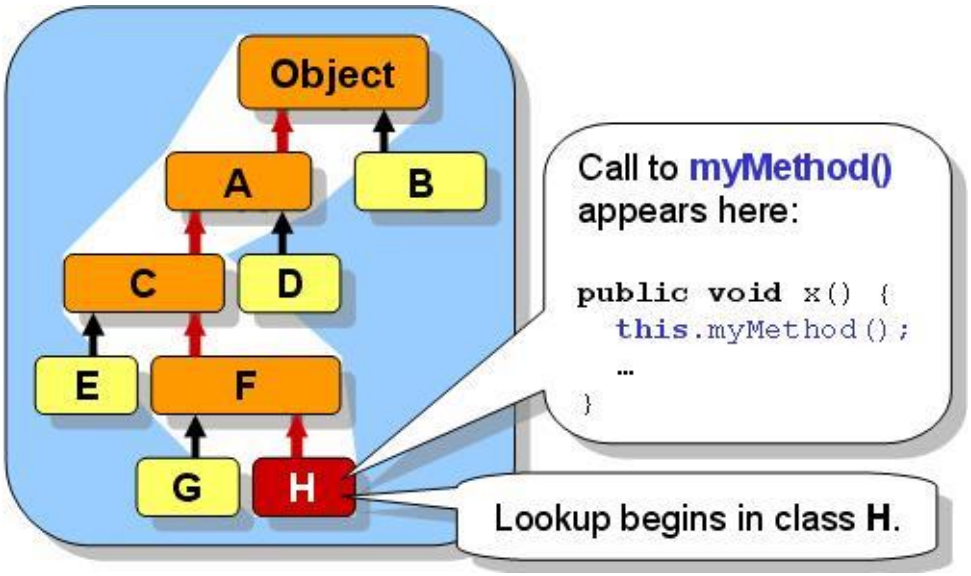


```
c.name = "Jim Clothes";
c.address = new Address();
c.phoneNumber = "1-613-555-5675";
c.itemsPurchased[0] = "Pencil Case";
c.purchaseHistory[0] = Date.today();

m.name = "Max E. Mumwage";
m.address = new Address();
m.phoneNumber = "1-613-555-8732";
m.employeeNumber = 232867;
m.hourlyPay = 8.75f;
m.duties[0] = "Phone Clients";
m.subordinates[0] = e;
```

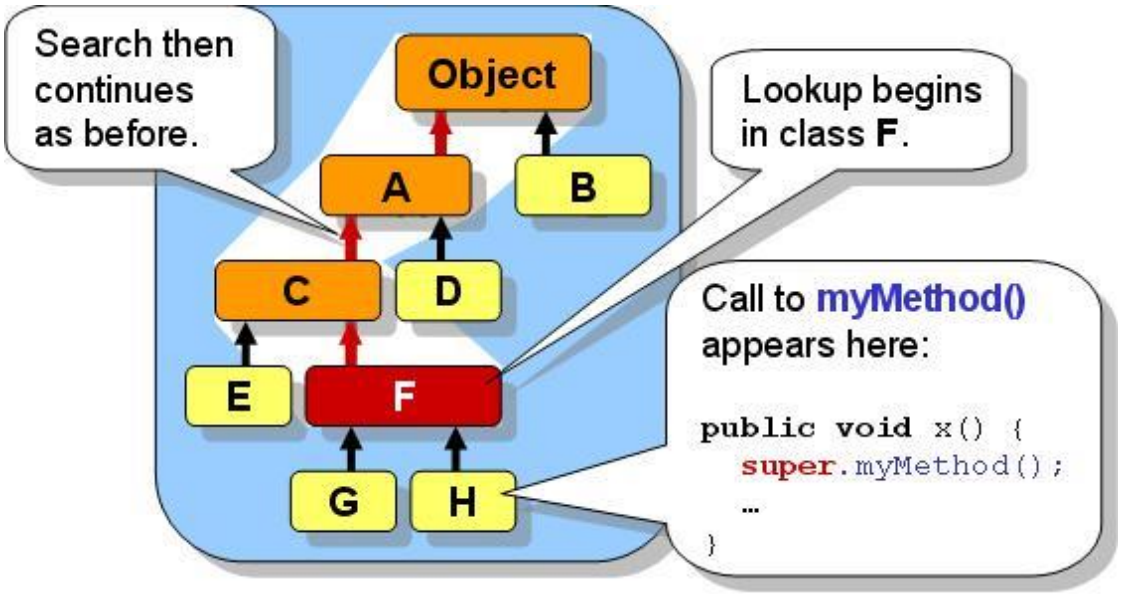
Own attributes



How Does Java Find Methods in the Hierarchy?



How Does Java Find Methods in the Hierarchy?

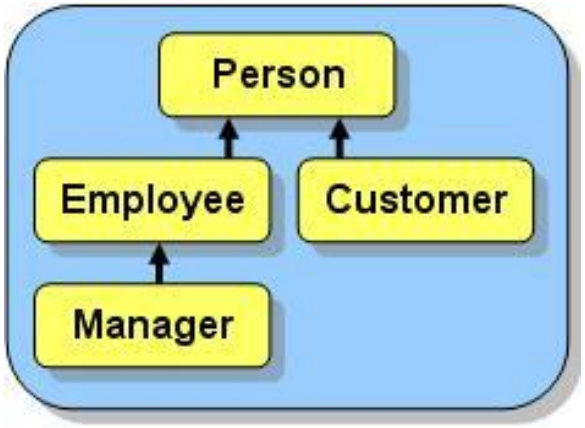


Access Modifiers

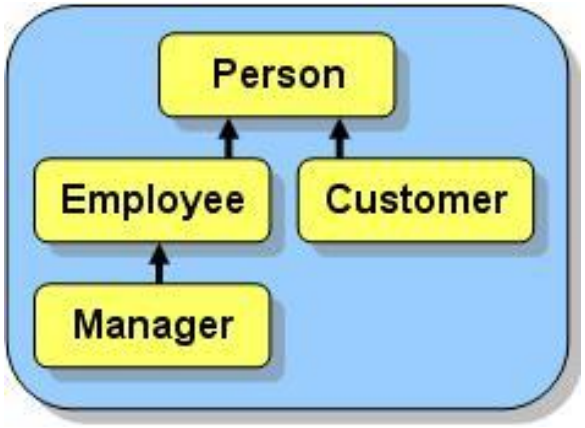
Access Modifiers

Private attributes and methods are inherited, but subclasses cannot access them.

```
public class Person
{
	private String name;
	private Address address;
	private String phoneNumber;
}
public class Employee extends Person
{
	private int employeeNumber;
	private float hourlyPay;
}
public class Customer extends Person
{
	private String[] itemsPurchased;
	private Date[] purchaseHistory;
}
public class Manager extends Employee
{
	private String[] duties;
	private Employee[]  subordinates;
}
```



```
public class Person
{
        private String name;
        private Address address;
        private String phoneNumber;
}
public class Employee extends Person
{
        private int employeeNumber;
        private float hourlyPay;
}
public class Customer ex
{
        private String[]
        private Date[] pu
}
public class Manager ext
{
        private String[] duties;
        private Employee[]  subordinates;
}
```




```
// Inside manager:
public boolean hasSeniority()
{
        return  (employeeNumber < 100)
          && (subordinates.length > 5);
}
```

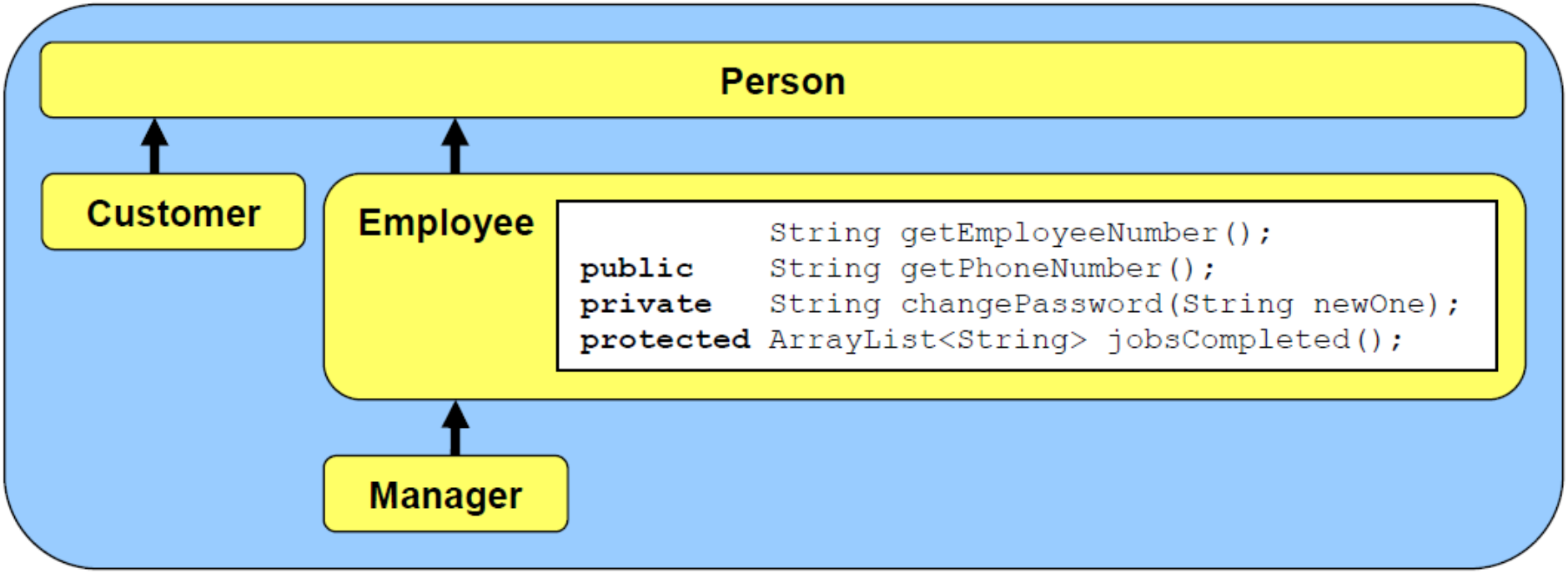
Compile error.

How can we solve this?

(1) use a public getter for the employee number
(2) declare attributes as protected instead

Access Modifiers

Protected attributes and methods are inherited, and subclasses *can* access them.


Person
Customer
Employee
String getEmployeeNumber();
public String getPhoneNumber();
private String changePassword(String newOne);
protected ArrayList<String> jobsCompleted();
Manager

```
public class Manager extends Employee
{
   public void tryThingsOut()
   {
        System.out.println(this.getEmployeeNumber());
        System.out.println(this.getPhoneNumber());
        System.out.println(this.changePassword("12345678"));
        System.out.println(this.jobsCompleted());
   }
}
```

```
public class Manager extends Employee
{
    public void tryThing
    {
        System.out.println(this.getEmployeeNumber());
        System.out.println(this.getPhoneNumber());
        System.out.println(this.changePassword("12345678"));
        System.out.println(this.jobsCompleted());
    }
}
```

Access allowed

```
public class Manager extends Employee
{
    public void tryThingsOut()
    {
          System.out.println
          System.out.println(this.getEmployeeNumber());
          System.out.println(this.changePassword("12345678"));
          System.out.println(this.jobsCompleted());
    }
}
```

Compiler error

```
public class Manager extends Employee
{
    public void tryThingsOut()
    {
          System.out.println(this.getEmployeeNumber());
          System.out.println(this.getPhoneNumber());
          System.out.println(this.changePassword("12345678"));
          System.out.println(this.jobsCompleted());
    }
}
```

Access allowed

```
public class Customer extends Person
{
   public void buyFrom(Employee emp)
   {
        System.out.println(emp.getEmployeeNumber());
        System.out.println(emp.getPhoneNumber
        System.out.println(emp.changePassword("12345678"));
        System.out.println(emp.jobsCompleted());
   }
}
```

```
public class Customer extends Person
{
    public void buyFrom(
    {
        System.out.println(emp.getEmployeeNumber());
        System.out.println(emp.getPhoneNumber
        System.out.println(emp.changePassword("12345678"));
        System.out.println(emp.jobsCompleted());
    }
}
```

Access allowed

```
public class Customer extends Person
{
    public void buyFrom(Employee emp)
    {
         System.out.println(emp.getEmployeeNumber());
         System.out.println(emp.getPhoneNumber
         System.out.println(emp.changePassword("12345678"));
         System.out.println(emp.jobsCompleted());
    }
}
```

Compiler error

```
public final void withdraw(float amount)
{
    ...
}
```

Nobody can override it

```
public final void withdraw(float amount)
{
     ...
}
```

Class Access Modifiers

```
// Accessible anywhere
public class Manager
{
    ...
}

// Accessible only inside a package
class Employee
{
    ...
}
```

Class Access Modifiers

```
// nobody can subclass it
public final class Manager
{
    ...
}
```