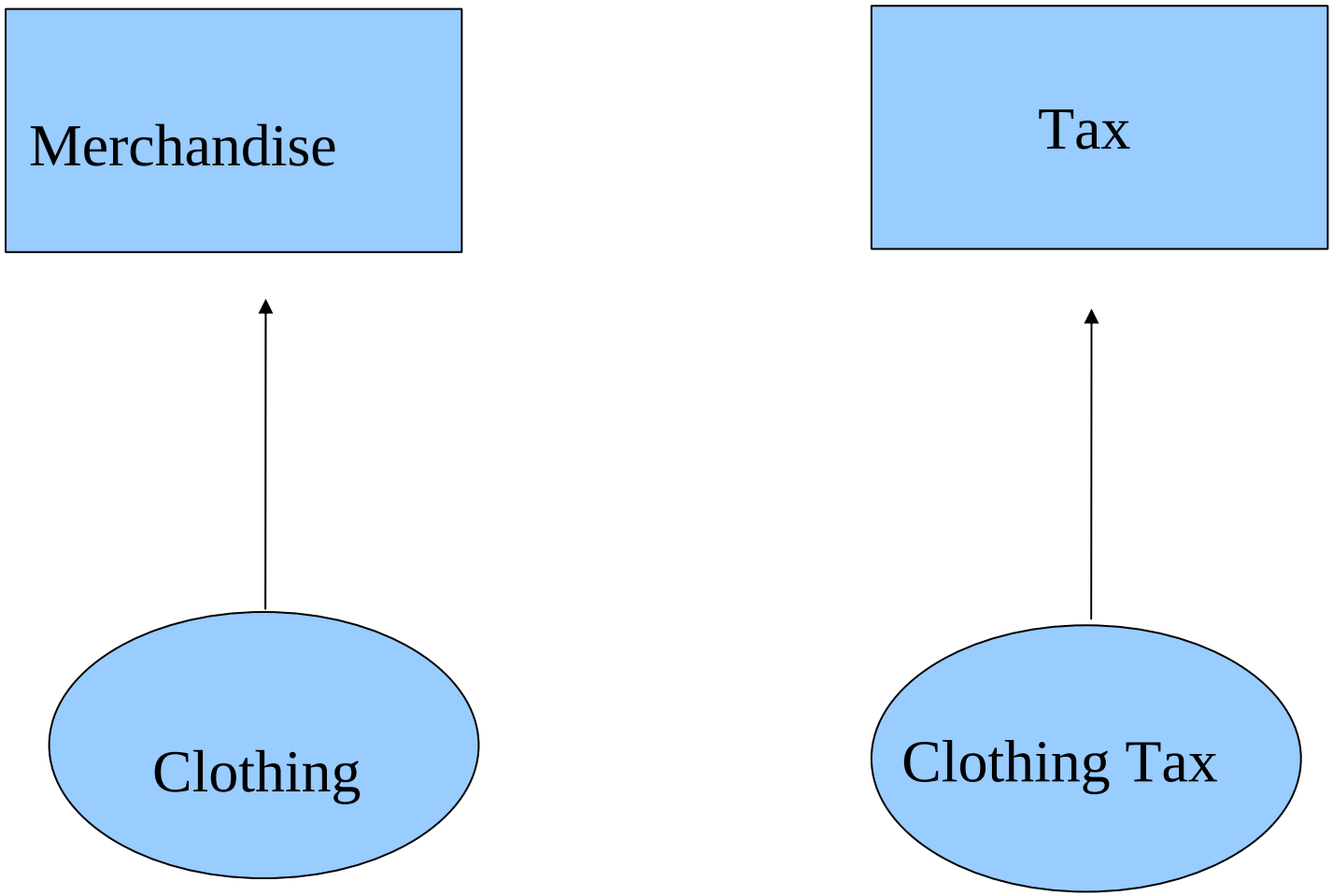
Class Hierarchy II

Discussion E

Hierarchy

A mail order business sells catalog merchandise all over the country. The rules for taxation on the merchandise vary from state to state. Also, the rate of taxation can be different based on the type of merchandise. For simplicity, we will consider three types of merchandise, clothing, pharmaceutical, and beauty products. Suggest a class hierarchy to model the tax on a merchandise.

Merchandise

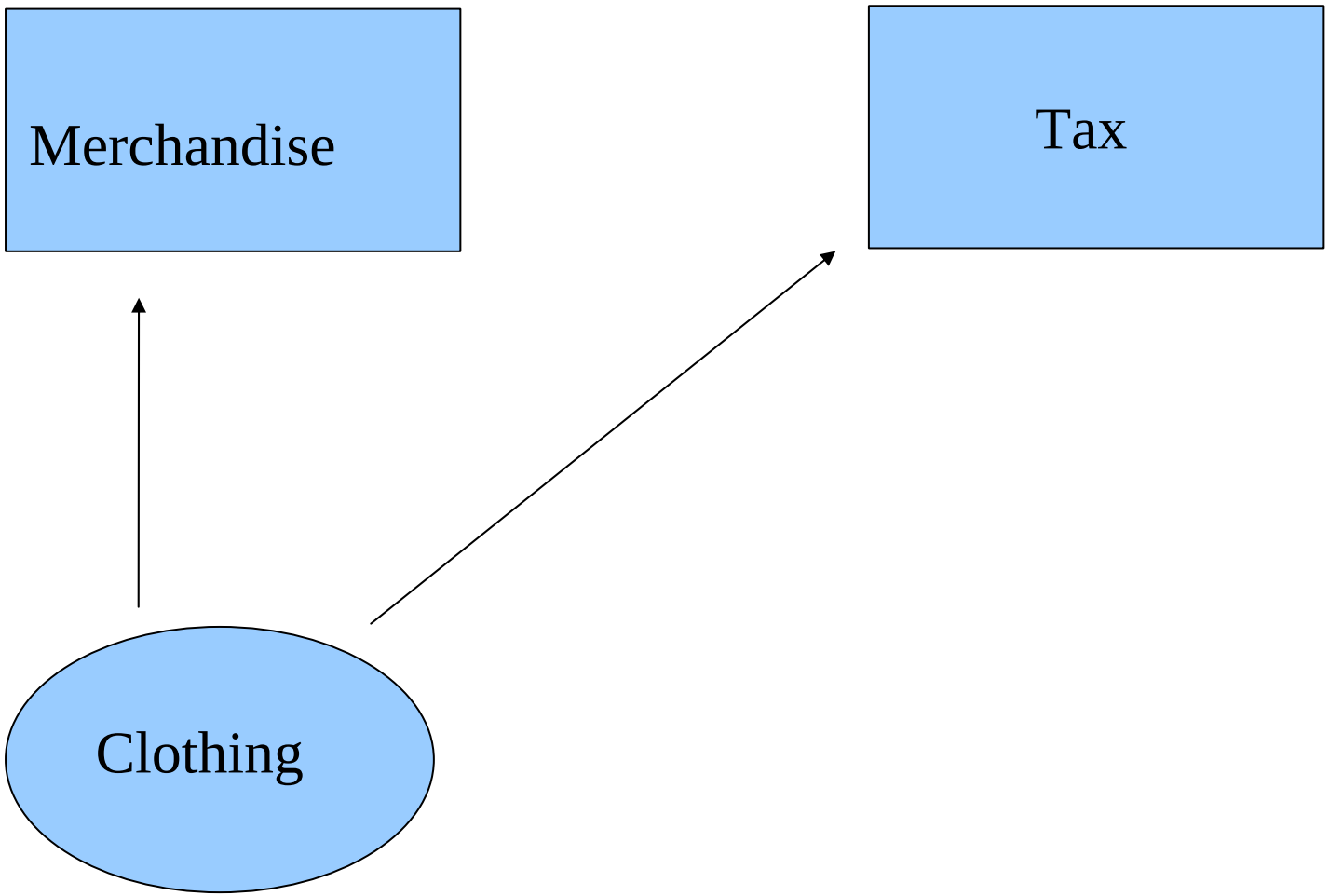


```
public abstract class Merchandise {

  Tax tax;
  public int getCost() {}

  public int getTax(int zipCode) {
    return tax.getTax(zipCode);
  }
}

public class Clothing extends Merchandise {

  public Clothing () {
    tax = new ClothingTax(this);
  }

  public int getCost() {}
}
```

```
public abstract class Tax {
 Merchandise article;
 public Tax();
 public int getTax(int zipCode);
}

public class ClothingTax extends Tax {

 //imagine a static zipcode indexed table for looking
 //up taxation

 public ClothingTax(Clothing article)
                    {this.article=article;}
 public int getTax(int zipCode);

}
```

We may want to model zip code explicitly using a Location class.

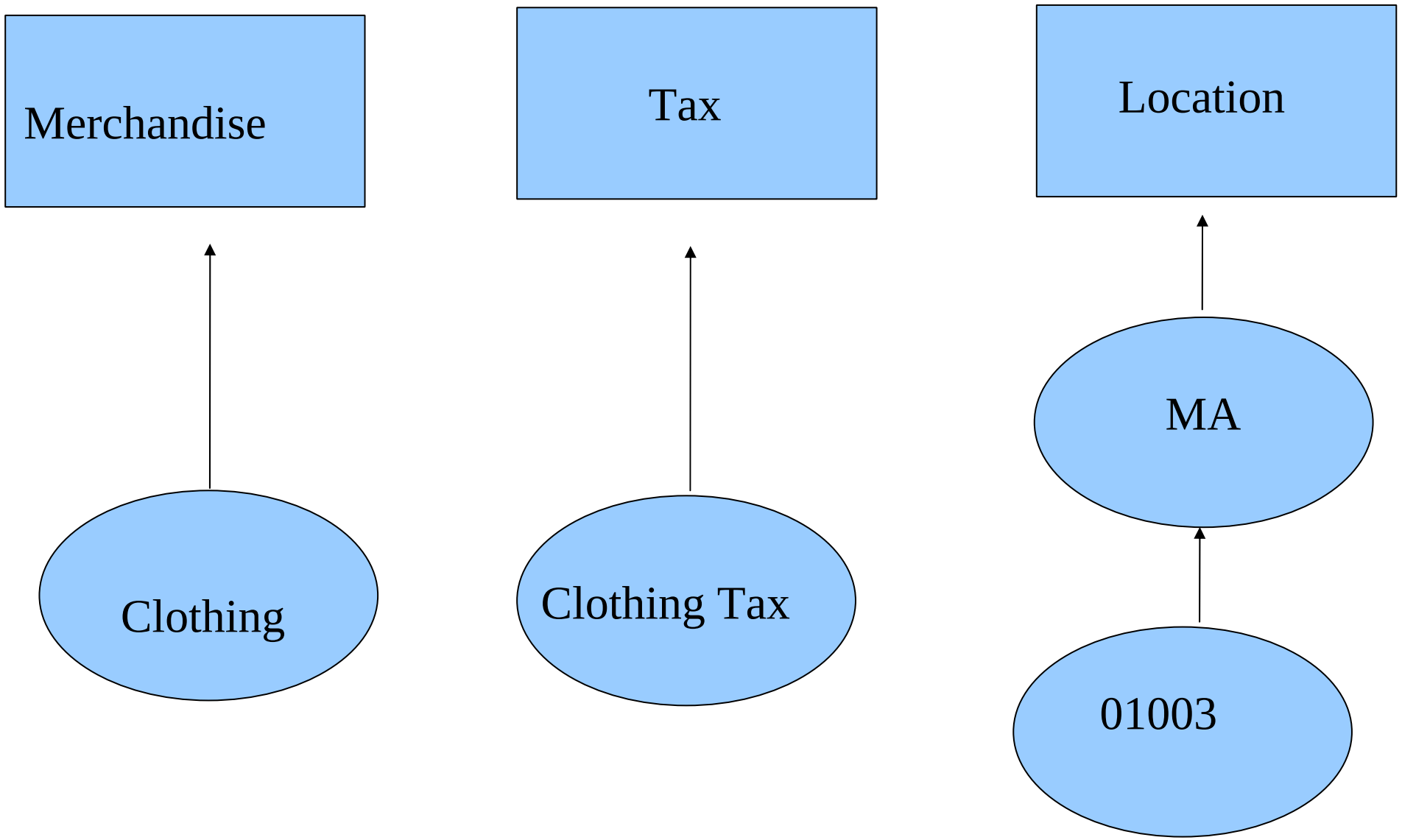
Interfaces



Extension

- assumed that tax rate was flat for a type
- it may depend on cost of item
  - clothes > $250 may be taxed differently

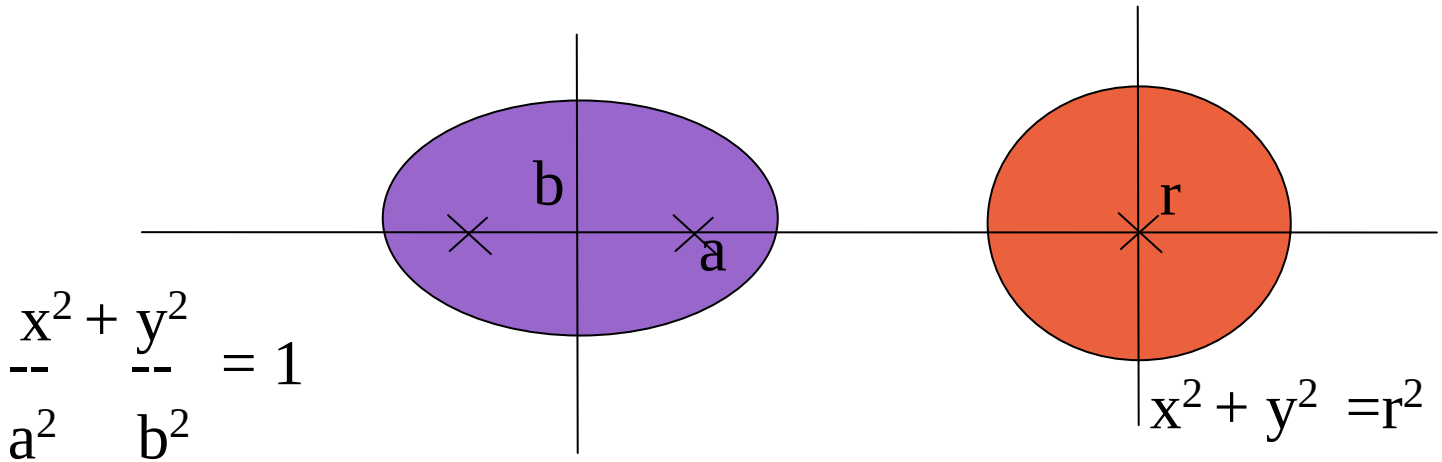
Detailed



Relationships

- Is-A
  - inheritence
- Has-A
  - composition
- Uses
  - parameters, calls
- Is-Used-By
  - Uses of other classes

Ellipse and Circle



$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

$$x^2 + y^2 = r^2$$

```
public class Ellipse
{
 int a;
 int b;
}
```

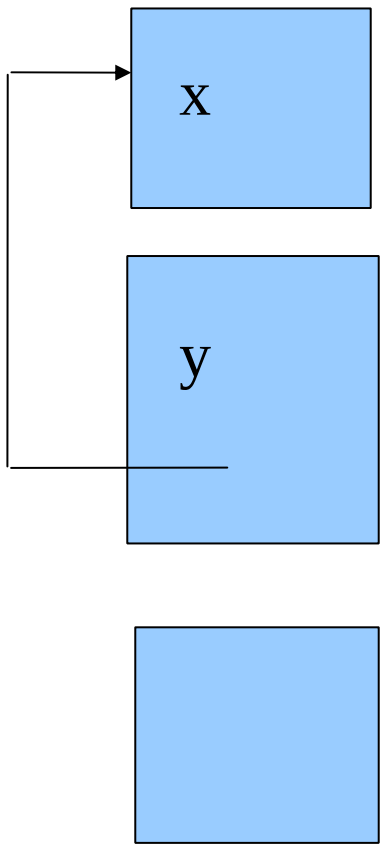
```
public class Circle
{
 int r;

}
```

Object Serialization

- Applications need to save data
- Saving an object may require saving of a sub-graph
- Object graph converted in series of bytes
- De-Serialization recreates objects from these bytes

Example (1)

```
public class X {
      int x;
}

public class Y {
      int y;
      X xobj;
}

public class App {
      Y yobj;
}
```



Y: y X: x

Serialization

- **`java.io.Serializable`**
- Marker interface, has no methods
- **`java.io.ObjectInputStream`**, **`java.io.ObjectOutputStream`** define default methods to read and write objects

Example (2)

```
public class X  implements  Serializable  {
  int x;
}

public class Y implements Serializable {
  int y;
  X xobj;
}

public class App {
  Y yobj;

  public void save() {}
  public void restore() {}

}
```

Example (3)

```
public void save {

  FileOutputStream fs = new FileOutputStream("y.save");
  ObjectOutputStream out = new ObjectOutputStream(fs);
  out.writeObject(yobj);
  out.close();

}
```

Example (4)

```
public void restore {

  FileInputStream fs = new FileInputStream("y.save");
  ObjectInputStream in = new ObjectInputStream(fs);
  yobj = (Y) in.readObject();
  in.close();

}
```

Safety

- Java generates a serialVersionUID for each class
- Matched for correctness at de-serialization
- You can override by defining you own field in class
  - **`static final long serialVersionUID = XXXX`**

Limitation

- No control on what gets written
  - Class info, fields stored as **<name, value>** pair
- Designed for generality
- Customization
  - **`readObject(),`** call **`defaultReadObject()`** first
  - **`writeObject(),`** call **`defaultWriteObject()`** first
- Optimization: fields marked **transient** are not serialized

Externalizable

- Lightweight
  - does not store name-value pair
  - **order is important,** serialization allows reading in any order
  - explicitly handle class hierarchy
  - no argument constructor needed
- Allows complete control on what gets written
- Methods defined in **`ObjectInput`** **`ObjectOutput`** can be used

Example (5)

```
public class X implements Externalizable {
  int x;
  void readExternal(ObjectInput in) throws IOEx,CNFEx{}
  void writeExternal(ObjectOutput out) throws IOEx {}
}

public class Y implements Externalizable {
  int y;
  X xobj;
  void readExternal(ObjectInput in) throws IOEx,CNFEx{}
  void writeExternal(ObjectOutput out)  throws IOEx {}
}
```

Example (6)

```
public class X  implements  Externalizable {

  int x;
  void readExternal(ObjectInput in) throws IOEx,
                                          CNFEx  {
    super.readExternal(in);
    x = in.readInt();
  }

  void writeExternal(ObjectOutput out) throws
                                          IOEx {
    super.writeExternal();
    out.writeInt(x);
  }
}
```

Example (7)

```
public class Y  implements  Externalizable  {
  int y;
  X xobj;

  void readExternal(ObjectInput in) throws IOEx, CNFEx{
    super.readExternal(in);
    y = in.readInt();
    xobj = in.readObject();
  }

  void writeExternal(ObjectOutput out) throws IOEx {
    super.writeExternal();
    out.writeInt(y);
    out.writeObject(xobj);
  }
}
```

Example (8)

```
public class App {
  Y yobj;

  public void save {
    FileOutputStream fs= new FileOutputStream("y.save")
    ObjectOutput out = new ObjectOutput(fs);
    yobj.writeExternal(out);
    out.close();
  }
  public void restore {
    FileInputStream fs = new FileInputStream("y.save");
    ObjectInput in = new ObjectInput(fs);
    Y nyobj = new Y();
    nyobj.readExternal(in);
    in.close();
  }
}
```