private int private;

private int private;

class MyInteger {
    
    // For example, is the following class definition allowed?

    - Can one object access the private and protected members (i.e., fields, methods, and constructors) of another object of the same class?

    Understanding Private and Protected
instance of a class.

instances of a class) rather than at the object level (this particular

* This is because access restrictions apply at the class or type level (all

protected members.

- Objects of the same type have access to one another's private and

• Yes.

Understanding Private and Protected (cont.)
public Apolyomial add(Apolynomial p)
    // asks p to add this Apolyomial to itself
    if (this.coefficients.length == 0) return p;
    if (p.coefficients.length == 0) return this;
    if (this.degree == 0) return p;
    if (p.degree == 0) return this;
    // returns a Apolyomial whose coefficients is the sum
    else
        return new Apolyomial();
}

class ConstantPoly extends Apolyomial

Example
2. I want to pass the object as an argument to a method.

1. I need to access a field that is obscured by a parameter or method is being performed. It's useful when you...

In any method or constructor, this refers to the object on which the

'this'
written so far unchanged?
touching any of the existing code, leaving everything that has been
is there a way to add new behavior to AList or APOJObject without
order to add methods to it.
Each time we want to compute something new, we edit each class in

What's Wrong With This Picture?
one more method to AllList.

* For AllList to execute any of these algorithms, we just need to add
  computations that we want AllList to perform.

- The variant behaviors are the infinitely many algorithms (i.e.
  and getRandom()).
- The invariant behaviors are the constructor and methods getRandom()

The key is to encapsulate the variant behaviors in a separate Union

... Toward a Solution ...
The Visitor Pattern

Host union.

It has a separate method for each of the concrete variants of the

- An abstract visitor is usually defined as an interface in Java.

The visitor pattern is a framework for communication and collaboration between two union patterns: a "host" union and a "visitor" union.
... {
    Object input;
    public abstract Object execute(Illustrate algo,
        ... 
    }
    public abstract class AList
    { ... 
        visitor ...
        The abstract host has a method (called the "hook") to "accept"
... {
    {
        return algo::forNonempty(this, input);
    }
    public Object execute(IList algo::Object algo::execute(input) {
    ...
}

class NEList extends ALList

visitor method: — and leaves it up to each of its concrete variants to call the appropriate
The Visitor Pattern (cont.)
public Object Object ForNonEmpty(Allist host, Object input)
{
    ...
    
    public Object Object ForNonEmpty(Allist host, Object input)
}
Visitors as well! If we have to modify the host union, then we will have to modify ALL... and does not change.

- This extensibility only works if the taxonomy of the host union is stable.
- Algorithms without changing any of the host union code.
- Algorithms on the host permits the addition of infinitely many external algorithms on the host's structural behaviors from the extrinsic dep

The Visitor Pattern (cont.)
Declaring Interfaces

A set of method and constant declarations without the method implementations.

Example

One interface can extend another interface:

\begin{verbatim}
\{ public int get\(A\)() \{
    return 1;
\}
\}
\end{verbatim}

Example

\begin{verbatim}
\{ public int get\(B\)() \{
    return 1;
\}
\}
\end{verbatim}

What is an interface?
Using Interfaces

- An interface is a reference type, just like a class.

```java
public class Colorable {
    int color;
}

public void setColor(int color) {
    this.color = color;
}

public int getColor() {
    return color;
}

class Color extends Point implements Colorable {
    // Example
    }

class Point { int x, y; }

// Example
```

- In a class definition, we say that a class implements an interface.

How do you use an interface?
Using Interfaces (cont.)

```java
public class YourInterface
{
    public static void setFinish(int finish)
    {
        finish = finish;
    }

    public static int getFinish()
    {
        return finish;
    }
}

A class can implement one or more interfaces.
```