Abstract Classes
Abstract Classes

- A class is **abstract** if it is not intended to be instantiated directly; rather, only objects in derived classes are instantiated.

- Each derived-class object **implicitly** entails an underlying base-class object.
Abstract Classes in Java

- In Java an abstract class is so-declared:
  ```java
  abstract class MyClass {
  ...
  }
  ```

- A class declared abstract cannot be instantiated directly.
Abstract Methods

- A *method* is *abstract* if there is no code for it in the abstract class; instead the meaning of the method is obtained from *over-riding* in derived classes.

- Only abstract classes can contain abstract methods.
Consider the following type of Tree:

- A Tree can be an Atom
- A Tree can be a Composite: a pair of Sub-Trees (each of which is a tree in its own right).

Type Tree is abstract:

- We never create a tree directly.
- We only create an Atom or a Composite.
Tree in Java

abstract class Tree
{
}

class Leaf extends Tree
{
    Object value;

    Leaf(Object value) {
        this.value = value;
    }
}

class Composite extends Tree
{
    Tree left;
    Tree right;

    Composite(Tree left, Tree right) {
        this.left = left;
        this.right = right;
    }
}
Adding an Abstract Method to Tree

- Add method

  ```
  int leafCount();
  ```

  to Tree
abstract class Tree
{
    public abstract int leafCount();
}
leafCount in the Derived Classes

class Leaf extends Tree
{
    Object value;

    Leaf(Object value)
    {
        this.value = value;
    }

    int leafCount()
    {
        return 1;
    }
}

class Composite extends Tree
{
    Tree left;
    Tree right;

    Composite(Tree left, Tree right)
    {
        this.left = left;
        this.right = right;
    }

    int leafCount()
    {
        return left.leafCount() + right.leafCount();
    }
}
Abstract Class vs. Interface

- An abstract class can contain data and method implementations; an interface cannot.

- It is common for an abstract class to define methods with the intention that they be overridden differently by each derived class.

- This is similar to different implementations of a common interface.
Multiple Inheritance

- Some languages allow one class to derive from multiple base classes; Java does not.

- The nearest thing would be a class deriving from a single basic class and implementing one or more interface at the same time.