Inheritance

AP Computer Programming III
Using Inheritance

- Create a new class by *extending* an existing class.
- The new class (subclass) *inherits* the data and methods of the existing class (superclass). (*code reuse*)
- The subclass can have additional data and methods.
- The subclass can *redefine* (override) inherited methods for different behavior.
Inheritance: An Example

**SlowFish:**

```java
public class SlowFish extends Fish {
    // don't declare inherited stuff
    // additional instance variable
double probOfMoving;

    // redefine nextLocation method
    protected Location nextLocation() {
        <new implementation>
    }
}
```
Inheritance: An Example

- SlowFish inherits some methods (e.g., move, changeLocation) and redefines some methods (e.g., nextLocation).
- An inherited method may call a redefined method (e.g., move calls nextLocation) unless it is private.
- A redefined method may call an inherited method (e.g., nextLocation calls changeLocation) unless it is private.
Inheritance: Another Example

- All classes extend the Object class (or are a subclass of Object).
- All classes inherit or redefine the equals and toString methods from Object.
- Any object may be put in an ArrayList (or other collection class) because they expect objects of type Object (and all objects are of type Object).
Inheritance: Another Example

**ArrayList:**

```java
ArrayList list = new ArrayList();
list.add(new Fish());
list.add(new DarterFish());
list.add(new Balloon());
for (int k = 0; k < list.length; k++)
{
    Object obj = list.get(k);
    System.out.println(obj.toString());
}
```

**Can only use Object methods with obj.**
Inheritance Supports …

✓ Generic Data Structures
  • Collection classes that act on any items of type Object; must cast to actual type

✓ Related Classes
  • shared data and behavior are inherited from shared superclass; some behavior may be redefined in subclass; additional behavior may be added
Inheritance: Key Ideas

- Inheritance provides a mechanism for code reuse as well as type flexibility and type-safe dynamic binding.
- Since subclasses are subtypes, subclasses should model the IS-A relationship (e.g., a DarterFish IS-A Fish; a Balloon is NOT a Fish).