

Interfaces

If a type is defined only by the existence of certain method headers – no storage of information, no method bodies – you can define an interface instead of an abstract class.

Why an interface instead of an abstract class?

- a class can implement multiple interfaces but can only extend one class
 - this avoids ambiguity about which method body to use should different bodies for the same method header be inherited from different places

CPSC 225: Intermediate Programming • Spring 2025

Multiple Inheritance

something may belong to more than one category – a bat (the creature) is both a kind of mammal and a kind of flying thing

```
to reflect this, we might want
public class Bat
   extends Mammal, FlyingThing {
    ...
}
```

but the specific semantics of extends creates some problems because classes, even abstract classes, can have instance variables and method bodies –

what if Mammal has an instance variable String type_ and FlyingThing has an instance variable int type_? Bat inherits both, but you can't have two variables with the same name and different types

Java avoids this problem by providing *interfaces* and not allowing multiple inheritance

CPSC 225: Intermediate Programming • Spring 2025

3

Interfaces - Defining

- interfaces address the desire for multiple inheritance
 - supports polymorphism but not code reuse
- syntax
 - public interface InterfaceName { ... }
 - public returntype methodname (paramlist);
 - like an abstract method in that no body is supplied, but abstract keyword is not needed
 - cannot have instance variables, constructors, or methods with bodies
- semantics
 - an interface defines a type
 - the type can be used anywhere a type is needed e.g. in variable and parameter declarations or as the base type of an array
 - it is not possible to create instances of an interface type
 - no new ... for an interface

13

Interfaces - Implementing

- classes implement interfaces
- syntax
 - public class ClassName implements Interface1, Interface2, ... { ... }
 - a class can implement any number of interfaces (it can also extend another class)
 - the class must provide a body for every method in the interface(s) or else it must be declared abstract
- semantics
 - an object of a type implementing an interface can be used anywhere the interface type is expected
 - e.g. Interface1 obj = new ClassName();

CPSC 225: Intermediate Programming • Spring 2025

Example

Things that are like eagles can fly and hunt. Things that are like lions can run and hunt.

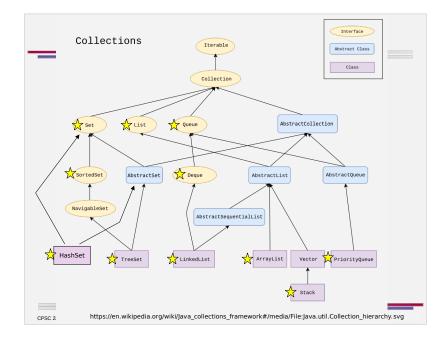
Eagles are, of course, eagle-like.

Lions are, of course, lion-like. They can also roar, which is not necessarily true of everything which is like a lion.

A griffin is like both eagles and lions. It can also guard treasure.

 define types (classes, abstract classes, and/or interfaces) to capture the relationships and behavior described

Abstract Classes What are the similarities and differences between an abstract class and an interface? vs Interfaces can have methods for [Choose] both which no body is defined] abstract can have methods for which a body is defined class can be used to neither [Choose] construct objects can define types (and both [Choose] thus can be used in variable declarations, parameter declarations, return types, and base types for arrays) can be [Choose] both extended/implemented by more than one class a single class can interface [Choose] extend/implement multiple CPSC 225: Intermediate Programming • Spring 2025



CPSC 225: Intermediate Programming • Spring 2025

