Interfaces

- an **interface** is a reference type in Java, stored in a .java file and can be seen as a set of only **abstract** methods

- thus, you cannot instantiate an interface and an interface does not contain any constructors

- when a class **implements** an **interface** it inherits all the (implicitly abstract) methods of the interface

- an interface may also contain constants, default and static methods, and nested types; implementations may exist only for static and default methods (more on this later)

- an interface **describes behaviours** that a class implements

- any class can implement multiple interfaces and an interface itself can extend multiple interfaces

Interface Example

Animal

void *makeNoise*()

<interface> Pet

void *cuddle*()

```
abstract class Animal {
  abstract void makeNoise();
}
```

Why do interfaces guarantee that there is no name clash?

```
interface Pet {
  void cuddle();   //implicitly abstract
  void makeNoise();//implicitly abstract
}
```

a class can inherit from one parent class and implement multiple interfaces as well

Dog

it is ok to have a method name identical to the one of another interface or even class defined in an interface since there is no implementation provided

for a Dog class to be a concrete (i.e. non-abstract) class it has to implement all abstract methods from all interfaces it implements and classes it extends

```
class Dog extends Animal implements Pet {

  @Override
  public void cuddle() { // from Pet
    System.out.println("Smelly cuddle!");
  }

  @Override
  public void makeNoise() { // from Animal
    System.out.println("Woof!");
} }
```

slide idea originally by N. Wu