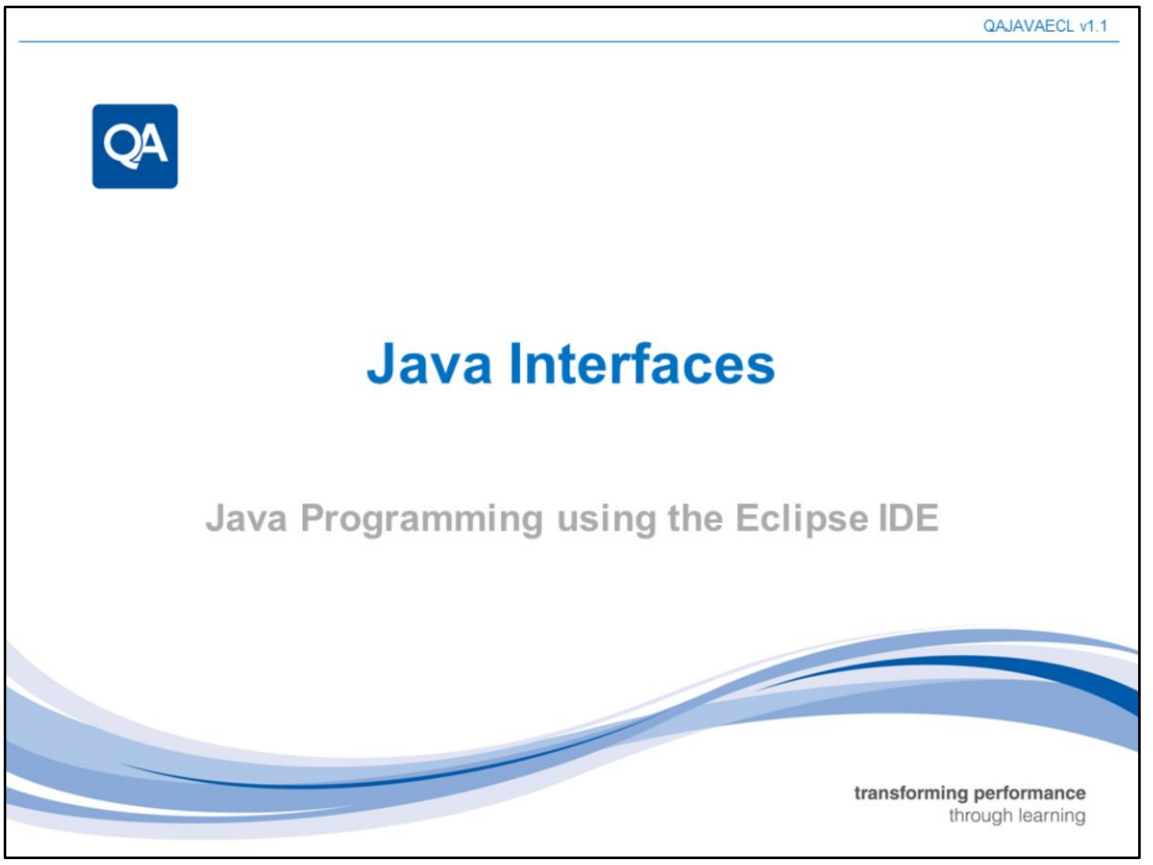

QAJAVAECL v1.1
Java Interfaces
Java Programming using the Eclipse IDE
transforming performance
through learning

Outline

- **Java Interfaces**
  - Implementing interfaces
  - Extending interfaces

Objectives

- **By the end of this session we should be able to**
  - Use interfaces to give objects particular sets of methods
  - Create an interface by extending another interface

Outline

- **Java Interfaces**
  - Implements
  - Extending interfaces

What is Java interface?

- **Java interface is a specification of method signatures**
- **Interface represents a pure abstract class concept**
- **Java only allows for single inheritance**
- **Interfaces are a way of ensuring an object has a set of behaviours without needing to inherit from more than one class**
  - Interfaces define method's name, return type and parameters required
  - Classes that implement an interface contain the method body and code
  - A class can implement more than one interface!
- **Fields in interfaces are implicitly public static and final**
  - Interfaces do not have their own state
  - You can't create an object from an interface

Implementing interfaces

- **Declare an interface using the interface keyword, rather than class**

```
public interface GuideDog {
        public String crossRoad();
        public boolean working();
}
```

- **Classes can then implement this interface**

```
public class Dog extends Animal implements GuideDog {

    @Override
    public String crossRoad() {
        //implementation here
    }

    @Override
    public boolean working() {
        //implementation here
    }
}
```

6

Extending interfaces

- **Interface can extend one or more interfaces at a time**

```
public interface GuideDog {
        public String crossRoad();
        public boolean working();
}
```

```
public interface RetiredGuideDog extends GuideDog{
        public String retirement();
        public boolean isRetired();
}
```

- **Class which implements RetiredGuideDog should override the methods of GuideDog and RetiredGuideDog otherwise the class itself should be declared as abstract**

7

Implementing child interface

```
public class Dog extends Animal implements RetiredGuideDog {
	@Override
	public String crossRoad() {
		//implementation here
	}
	@Override
	public boolean working() {
		//implementation here
	}
	@Override
	public String retirement(){
	  //implementation here
	}
	 @Override
	public boolean isRetired(){
	  //implementation here
	 }
   }
```

Extends vs Implements

- **Extends is object inheritance**
  - Object exists in a hierarchy
  - Can only extend from one class
  - Contain abstract method, concrete methods and fields
  - Explains what an object is

- **Implements uses interfaces**
  - Can implement many interfaces
    - Theoretical maximum of 65535
  - Contain unimplemented methods only
  - No internal state
  - Describe what an object should do

What class am I?

- **Each object knows what class it is when it is created**
  - We can access this using the getClass() method

```
Dog d = new Dog("Spot", 2);

System.out.println(d.getClass()); // class com.other.Dog
```

- **We can also see if an object is an instanceof another class**
  - This also works for checking if the class implements a particular interface

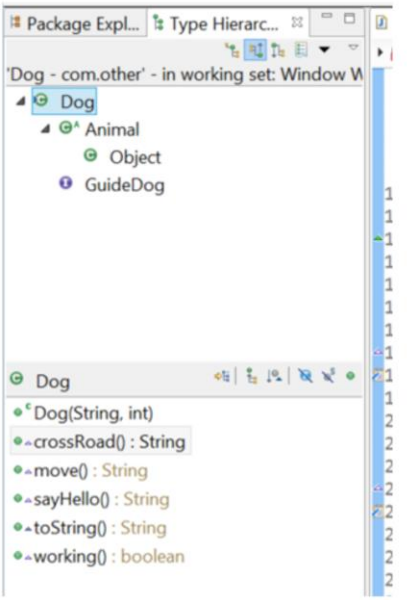
```
if (d instanceof Dog){
      System.out.println("It's a dog"); // It's a dog
}

if (d instanceof GuideDog) {
      // It's a Guide Dog!
      System.out.println("It's a Guide Dog!");
}
```

10

"instanceof" isn't a typo, this is the keyword used.

The type hierarchy in Eclipse

- **Eclipse can visualise how classes and interfaces are connected to one another**
- **Quick type hierarchy:**
  - Right click on the class declaration
  - Ctrl + T



11

Outline

- **Interfaces**
  - Implementing interfaces
  - Extending interfaces

Exercise

- Use interfaces to give the objects extra functionality