Unit objectives

- After completing this unit, you should be able to:
  - Explain the concept of an interface and outline what it provides
  - Declare an interface in Java
  - Declare that a class implements one or more interfaces
  - Explain what a class implementing an interface must supply
  - Identify programming scenarios in which interfaces are useful

Interfaces

- Declared types in Java are either classes or interfaces
  - Interfaces represent a promise of supported services of the objects which implement the interface
- An interface represents a contract that objects implement
  - An interface is simply a list of method declarations
- Its methods are declared but not implemented
- An interface is not a class, although it looks remarkably like an abstract class

```
public interface File {
    public void open(String name);
    public void close();
}
```

Implementing interface methods

- Methods declared in an interface are implemented in the classes which support that interface

```
public class TextFile implements File {
    public void open(String name) {
        // implementation of open method
    }
    public void close() {
        // implementation of close method
    }
}
```

Syntax

- In a class declaration, the naming of the superclass precedes any interfaces supported by the class

```
public class Directory extends Secure
             implements File {
        ...
}
```

- If a class implements multiple interfaces, the interfaces are all listed, separated by commas

```
public class Directory
        implements File, Secure {
        ...
}
```

Implementing an interface

- A class which implements an interface must implement every method defined by that interface
  - If one or more methods are not implemented, the compiler will generate an error
- Subclasses automatically implement all interfaces that their superclass implements
- Some interfaces, such as **Cloneable, do not contain any** methods
  - These act as a signal to Java that a class observes a certain protocol that may not be expressible as a specific set of methods
  - In the case of **Cloneable, it alerts the JVM that objects of** the implementing class can be copied via the clone() method

Typing and interfaces

- A variable's type can be an interface
  - Only objects whose class implements that interface can be bound to that variable
  - Only messages defined by the interface can be used
  - Interfaces cannot appear in a new expression

```
File r = new File(); // Error
File f = new TextFile(); // OK!
```

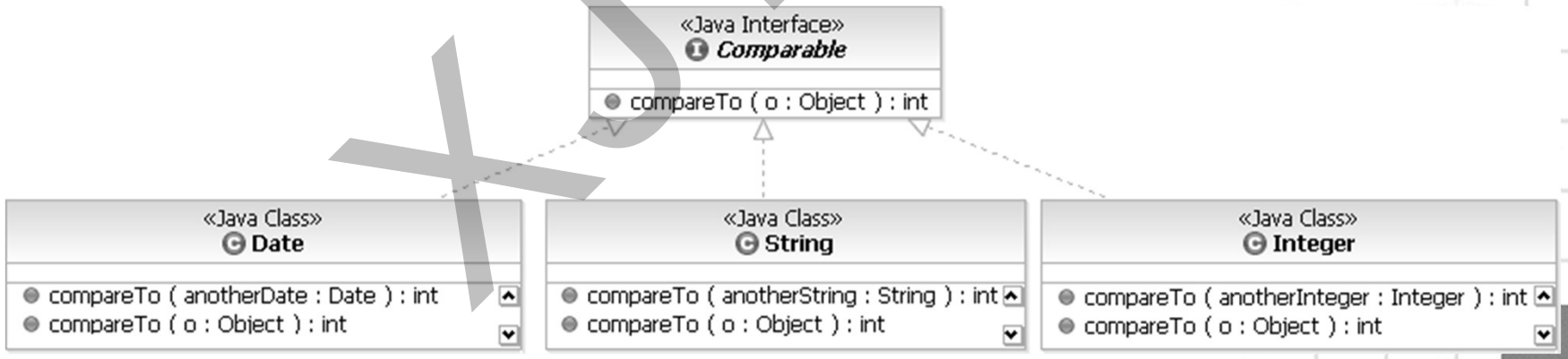
Subinterfaces

- Interfaces can be extended
  - Interfaces are not classes, so this hierarchy is independent of the class hierarchy
  - The interface which extends another interface inherits all of its method declarations

```
interface File {
   public void open(String name);
   public void close();
}

interface ReadableFile extends File {
   public byte readByte();
}

interface WritableFile extends File {
   public void writeByte(byte b);
}

interface ReadWriteFile extends ReadableFile,
                                 WritableFile {
   public void seek(int position);
}
```

Protocols

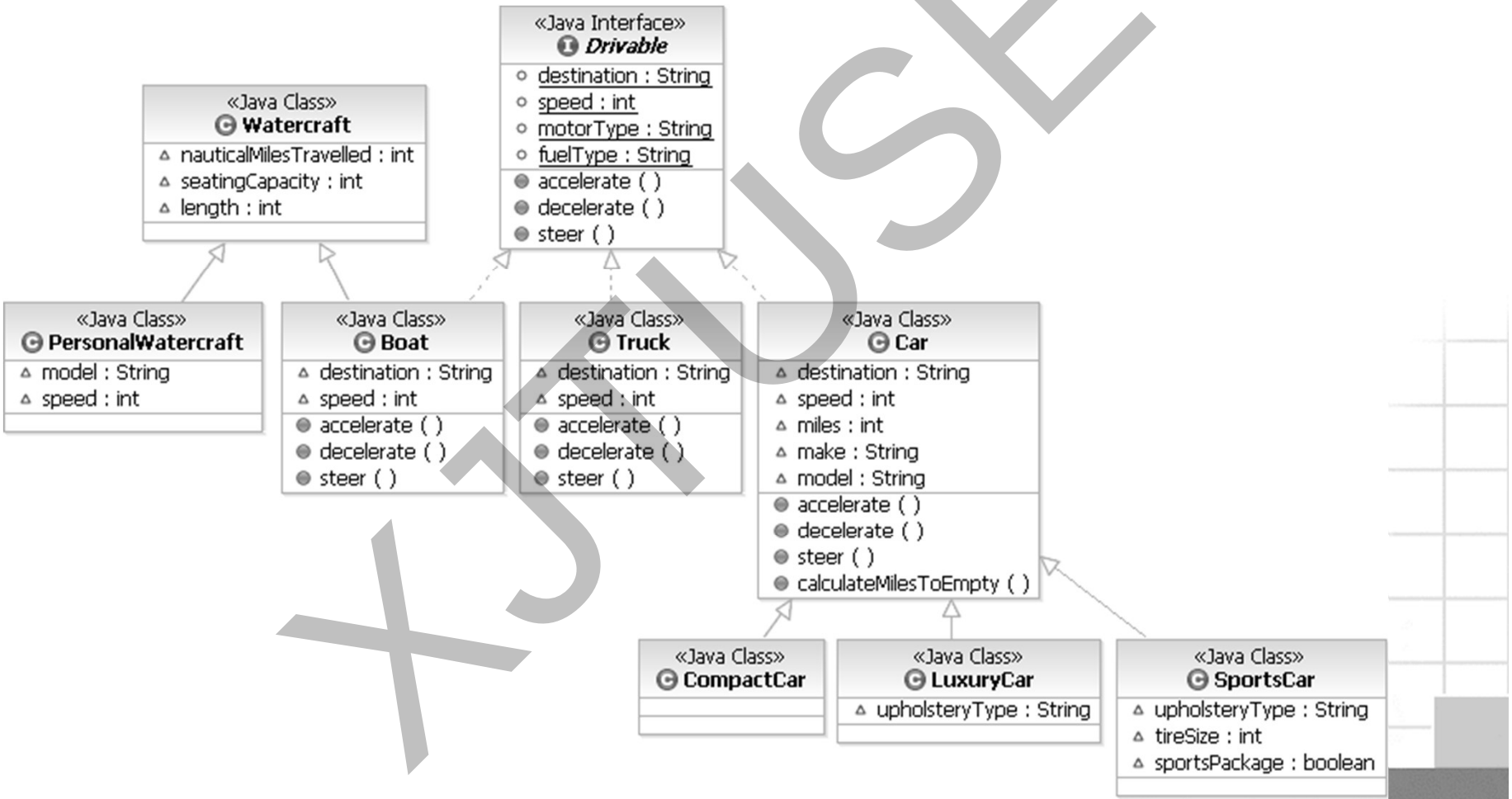
- An interface defines a protocol (a set of methods)
  - If a class implements a given interface, then that class implements that protocol
- An interface can be used to impose a common protocol on a group of classes that are not related by inheritance
  - In the chain of classes related by inheritance, the common protocol is imposed through subclassing



Using interfaces

- Cross-hierarchy polymorphism can be accomplished using interfaces
- Interfaces allow access to methods in separate class trees
  - Can substitute an object for another object which is not related via the class hierarchy
  - Classes that implement the same interface understand the same messages, regardless of their location in the class hierarchy

Example



More advantages of using interfaces

- Using interfaces allows more control over how objects are used
  - Similarities among classes not related by inheritance can be captured, without forcing an unnatural relationship
  - The developer knows which message the objects will respond to; an object's interface may be revealed without revealing its class
  - Improves reusability of code

Naming conventions for interfaces

- Make "able" interfaces
  - Cloneable, Serializable, and so on
- Name interfaces using proper nouns and provide "Impl“ implementation of your interfaces
  - **Bank interface, BankImpl class**
  - **BankAccount interface, BankAccountImpl class**
  - With this convention, the interface typically contains a definition for all (or most) of the implementation class’s public methods
- Prefix interface names with "I" and use proper nouns for your classes
  - **IBank interface, Bank class**
  - **IBankAccount interface, BankAccount class**

Checkpoint

- 1. What is the difference between an interface and a class?
- 2. How are interfaces like abstract classes? How are they not?

Unit summary

- In this unit, you should have learned to:
  - Explain the concept of an interface and outline what it provides
  - Declare an interface in Java
  - Declare that a class implements one or more interfaces
  - Explain what a class implementing an interface must supply
  - Identify programming scenarios in which interfaces are useful