Now consider this code fragment, in which **Bicycle** is also a subclass of **Vehicle**:

```
Vehicle v;
Car c;
Bicycle b;
c = new Car();
v = c; // okay
b = (Bicycle) c; // compile time error!
b = (Bicycle) v; // runtime error!
```

The last two assignments will both fail. The attempt to assign **c** to **b** (even with the cast) will be a compile-time error. The compiler notices that **Car** and **Bicycle** do not form a subtype/supertype relationship, so **c** can never hold a **Bicycle** object—the assignment could never work.

The attempt to assign **v** to **b** (with the cast) will be accepted at compile time, but will fail at runtime. **Vehicle** is a superclass of **Bicycle**, and thus **v** can potentially hold a **Bicycle** object. At runtime, however, it turns out that the object in **v** is not a **Bicycle** but a **Car**, and the program will terminate prematurely.

Casting should be avoided wherever possible, because it can lead to runtime errors, and that is clearly something we do not want. The compiler cannot help us to ensure correctness in this case.

In practice, casting is very rarely needed in a well-structured, object-oriented program. In almost all cases, when you use a cast in your code, you could restructure your code to avoid this cast and end up with a better-designed program. This usually involves replacing the cast with a polymorphic method call (more about this will be covered in the next chapter).

10.8 The `Object` class

Concept

All classes with no explicit superclass have **Object** as their superclass.

All classes have a superclass. So far, it has appeared as if most classes we have seen do not have a superclass. In fact, while we can declare an explicit superclass for a class, all classes that have no superclass declaration implicitly inherit from a class called **Object**.

Object is a class from the Java standard library that serves as a superclass for all objects. Writing a class declaration such as

```
public class Person
{
    . . .
}
```

is equivalent to writing

```
public class Person extends Object
{
    . . .
}
```