SE 2AA4 Winter 2007

06 Java Notes

William M. Farmer

Department of Computing and Software
McMaster University

28 February 2007

Note 1: Classes vs. Their Types

- Each expression has a unique type, and each object belongs to a unique class.
- An object O is an instance of a class C if C is O's class or C is a superclass of O's class.
  - Every object is an instance of the class `Object` since `Object` is a superclass of every other class.
- An object O is of a class type C if O is an instance of the class C.
- Type is a compile-time notion, while class is a run-time notion.
- The following cannot be deduced at compile-time since the class of an object may not be known until run-time:
  - Which method to invoke.
  - The outcome of a call to `instanceof`.
  - Correctness of a casting (explicit type conversion).

Note 2: Interfaces

- A Java interface consists of a set of public constant and method declarations.
  - The interface name becomes a new reference type.
  - An interface has no implementation at all.
  - An interface cannot be instantiated.
- Interfaces are good for recording similarities between unrelated classes.
- Interfaces cannot be extended (unlike classes).
- An object *O* is of an interface type *I* if *O*'s class implements the interface *I*.
- Key benefit of interfaces: If *I* is an interface, then a variable of type *I* may be bound to any object whose class implements *I*.

Note 3: Constants

- A constant should be specified as an immutable, initialized class or interface field, e.g., as:

```
final static int ZERO = 0;
```

- A global constant should be specified a public, immutable, initialized class or interface field, e.g., as:

```
public final static int ZERO = 0;
```

- The name of a constant should be written in uppercase, e.g., as:
  - `ZERO`.
  - `ZERO_VECTOR`.
- Interfaces are very convenient for managing groups of global constants.