SWEN20003
Object Oriented Software Development

Interfaces and Polymorphism Questions

Bach Le
bach.le@unimelb.edu.au

University of Melbourne
© University of Melbourne 2023

The Road So Far

- Subject Introduction
- Java Introduction
- Classes and Objects
- Arrays and Strings
- Input and Output
- Software Tools
- Inheritance and Polymorphism

Learning Outcomes

Upon completion of this topic you will be able to:

- Describe the purpose and use of an **interface**
- Describe what it means for a class to **use** an interface
- Describe when it is appropriate to use inheritance vs. interfaces
- Use interfaces and inheritance to achieve powerful abstractions
- Make any class "sortable"

Interfaces

Keyword

Abstract Class: A class that represents common attributes and methods of its subclasses, but that is **missing** some information specific to its subclasses. Cannot be instantiated.

Keyword

Interface: Declares a set of constants and/or methods that define the **behaviour** of an object.

Which one of these keywords is used by a class to use an interface defined previously?

1. extends
2. import
3. implements
4. implement
5. none of the above

Answer: 3

What type of a relationship motivates using an interface?

1. is-a
2. can-do
3. has-a
4. all of the above
5. none of the above

Answer: 2

Which of the following statements are true?

- A concrete class that implements the interface must implement at least one method in the interface.
- Interfaces can have instance variables.
- Interfaces can have default method implementations.
- A class that does not implement all methods in an interface must be abstract.
- Interfaces can be extended similar to classes.

Answer: 3, 4, 5

Answer true or false:

An interface that has some default implementations is analogous to an abstract class.

false

A class can implement multiple interfaces.

true

Java supports multiple inheritance by allowing implementing multiple interfaces.

false

The keyword `abstract` is used to define methods that do not have an implementation in an interface.

How does the `Arrays.sort(arrayOfThings)` method able to sort an array of similar objects?

Answer:
If the objects implement the comparable interface, with the `compareTo` method comparing the objects based on how it should be sorted, the `sort` algorithm can perform the sort using the comparable objects.

What is the signature of the `compareTo` method of the Comparable interface?

Answer:

```
public int compareTo(<ClassName> object)
```

How would you decide whether a class should inherit a class or implement an interface?

Inheritance is for generalising **shared properties** between **similar classes**; "is a".

Assess Yourself

Interface or Inheritance?

- All Dogs can bark.

Both?

Needs more context...

Assess Yourself

Interface or Inheritance?

- All `Animals`, including `Dogs` and `Cats` can make noise.

Inheritance

Assess Yourself

Interface or Inheritance?

- All `Animals` and `Vehicles` can make noise.

Interface

Assess Yourself

Interface or Inheritance?

- All classes can be compared with themselves.

Interface

Assess Yourself

Interface or Inheritance?

- All `Characters` in a game can talk to the `Player`.

Inheritance

Assess Yourself

Interface or Inheritance?

- Some `GameObjects` can move, some can talk, some can be opened, and some can attack.

Interface

Interface or Inheritance?

Inheritance:

- Represents *passing shared information* from a *parent* to a *child*
- Fundamentally an "Is a" relationship; a *child* **is a** *parent*, plus more; hierarchical relationship
- All `Dogs` are `Animals`

Interface:

- Represents the ability of a *class* to *perform an action*
- Fundamentally a "Can do" relationship; a *Comparable* object can be *compared* when sorting
- `Strings` can be compared and sorted

Metrics

A `Student` is specified by a first and last name, a student ID, and a list of subjects. When `Students` are sorted, they should appear in increasing student number order.

A `Subject` is specified by a name, subject code, and a list of students. When `Subjects` are sorted, they should appear in order of ascending subject code.

A `Course` is specified by a name, a course code, a list of (possible) subjects, and a list of students. When `Courses` are sorted, they should appear in order of ascending course code.

Implement appropriate `compareTo` methods for each class, and implement the `Enrollable` interface such that a `Student` can enrol in both a `Subject` and a `Course`.

Lecture Objectives

After this lecture you will be able to:

- Describe the purpose and use of an **interface**
- Describe what it means for a class to **use** an interface
- Describe when it is appropriate to use inheritance vs. interfaces
- Use interfaces and inheritance to achieve powerful abstractions
- Make any class "sortable"

The Road So Far

- Subject Introduction
- Java Introduction
- Classes and Objects
- Arrays and Strings
- Software Tools
- Input and Output
- Inheritance and Polymorphism