CSE 115 / 503
INTRODUCTION TO COMPUTER SCIENCE I

Dr. Carl Alphonce
Dr. Jesse Hartloff

University at Buffalo
School of Engineering and Applied Sciences

CSE 50
1967-2017

Survey Announcement

- To help us better understand different aspects of how this course is working we will administer three on-line surveys.

- Incentive: everyone will earn 1 Proficiency Point for each survey, but only if overall class particiation goes over 80% for that survey.

Interfaces

Defining the word ‘interface’

interface |ˈin(t)ərˌfās|

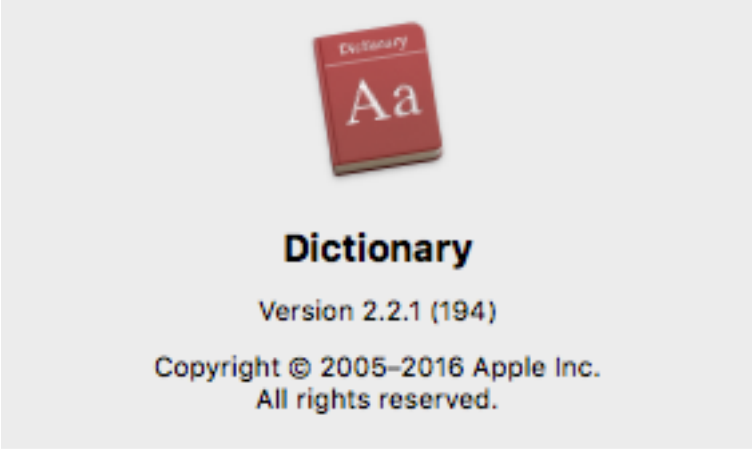
noun

1 a point where two systems, subjects, organizations, etc., meet and interact: *the **interface between** accountancy and the law*.

• *chiefly Physics* a surface forming a common boundary between two portions of matter or space, e.g., between two immiscible liquids: *the surface tension of a liquid at its air/liquid interface*.

2 *Computing* a device or program enabling a user to communicate with a computer.

• a device or program for connecting two items of hardware or software so that they can be operated jointly or communicate with each other.



Defining the word ‘interface’ (as in ‘Graphical User Interface’)

interface |ˈin(t)ərˌfās|

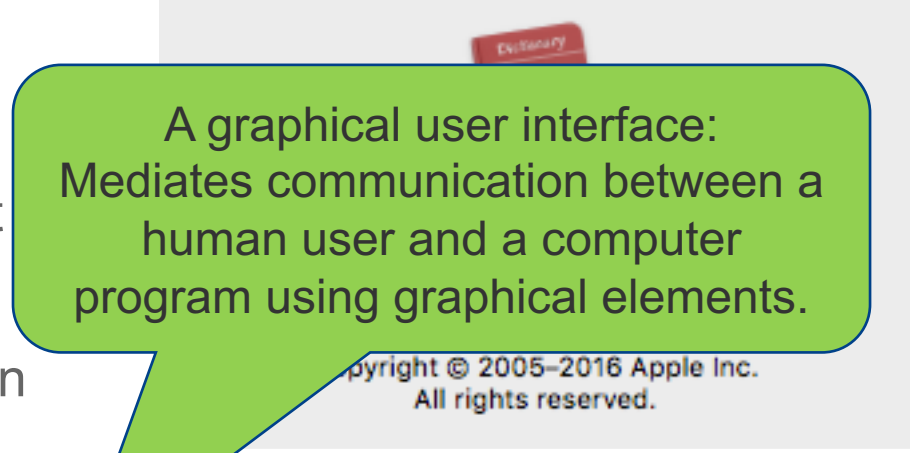
noun

1 a point where two systems, subjects, organizations, etc., meet interact: *the **interface between** accountancy and the law*.

• *chiefly Physics* a surface forming a common boundary between two portions of matter or space, e.g., between two immiscible liquids: *the surface tension of a liquid at its air/liquid interface*.

2 *Computing* **a device or program enabling a user to communicate with a computer**.

• a device or program for connecting two items of hardware or software so that they can be operated jointly or communicate with each other.



A graphical user interface: Mediates communication between a human user and a computer program using graphical elements.

Defining the word ‘interface’ (as in a Java ‘interface’)

interface |ˈin(t)ərˌfās|

noun

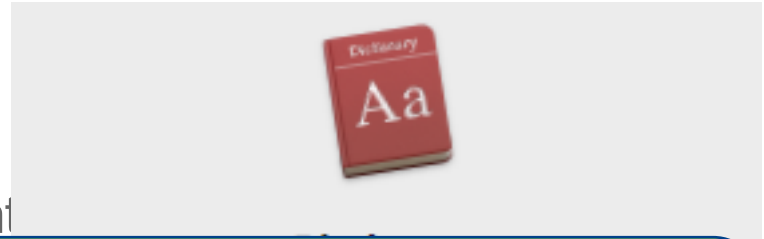
1 a point where two systems, subjects, organizations, etc., meet and int

*the **interface between** accountancy and the law*.

• *chiefly Physics* a surface forming a common boundary between two portions of matter or space, e.g., between two immiscible liquids: *the surface tension of a liquid at its air/liquid interface*.

2 *Computing* a device or program enabling a user to communicate with a computer.

• a device or **program for connecting two items of** hardware or **software so that they can** be operated jointly or **communicate with each other**.



A Java interface specifies how two one software component can interact with another software component.

SYNTAX: form of an interface

header + body

header

- access control modifier
- keyword ‘interface’
- name (generally an adjective, following class name conventions, but prefixed with an upper-case ‘`I`’)

body

- method specifications (method headers followed by ‘;’, also called method declarations, as opposed to method definition)
- a few other things are permitted in interfaces (e.g. Java 8 now allows “default methods”) we won’t worry about these right now.

Examples from Java's libraries (small detail omitted)

```
public interface ActionListener {
    public void actionPerformed(ActionEvent e);
}


public interface Runnable {
    public void run();
}


public interface MenuKeyListener {
    void menuKeyTyped(MenuKeyEvent e);
    void menuKeyPressed(MenuKeyEvent e);
    void menuKeyReleased(MenuKeyEvent e);
}
```

Interfaces – no instantiation

While classes can be instantiated, interfaces cannot be instantiated.

Why is this?

Realization

Realization a.k.a. Implementation

Realization is a relationship between a *class* and an *interface*.

An interface contains *method specifications*, rather than full method definitions. A method specification is also called an *abstract method* or a *method declaration*.

A class contains *concrete methods*, a.k.a. *method definitions*.

Implementation - 1

A class can *implement* an interface:

```
public class EventHandler implements ActionListener {
   ...
}
```

Implementation as contract

A class which implements an interface is obligated to provide full definitions of all the methods specified in the interface.

Implementation - 2

```
public class EventHandler implements ActionListener {
    ...
    @Override
    public void actionPerformed(ActionEvent e) {
        ...
    }
    ...
}
```

Implementation - 2

```
public class EventHandler implements ActionListener {
    ...
    @Override
    public void actionPerformed(ActionEvent e) {
        ...
    }
    ...
}
```

@Override indicates that the method to which is is applied implements a method specified in an interface.

Concrete example

```
public class EventHandler implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
                System.out.println("Button clicked");
        }

}
```

Types

When you define a class, you are defining a type.

When you define an interface, you are also defining a type.

A class which implements an interface is a SUBTYPE of the interface type.

An instance of the class belongs to both types:
the type that the class defines AND
the type that the interface defines.

An instance of EventHandler belong to two types:

EventHandler
ActionListener

```
public class EventHandler implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
                System.out.println("Button clicked");
        }

}
```

Assignment

If a variable is declared to be of an interface type (e.g. IType), it can be assigned an instance of any subtype class (e.g. C1):

```
public class C1 implements IType {…}
public class C2 implements IType {…}

IType var;
var = new C1 (); // C1 is a subtype of IType
var = new C2 (); // C2 is a subtype of IType
```

Point of variation

If a variable is declared to be of an interface type (e.g. IType), it can be assigned an instance of any subtype class (e.g. C1):

```
public class C1 implements IType {…}
public class C2 implements IType {…}

IType var;
var = new C1 (); // C1 is a subtype of IType
var = new C2 (); // C2 is a subtype of IType
```

var is a point of variation in the code (more to come...)