CSE115 / CSE503
Introduction to Computer Science I

Dr. Carl Alphonce
343 Davis Hall
alphonce@buffalo.edu

Office hours:
Thursday 12:00 PM – 2:00 PM
Friday 8:30 AM – 10:30 AM

OR request appointment via e-mail
Turn off and put away electronics:

- cell phones
- pagers
- laptops
- tablets
- etc.
Last time

- class – class inheritance
- interface – interface inheritance
- the class Object

Coming up

- overloading
- constructor chaining

- primitive types in more detail
Exam #2

Tuesday 4/12, 9:15 PM - 10:15 PM

Abayev – Dawkins  NSC 201
De Jesus – Geiger  NSC 210
Gensel – Jaiswal  NSC 215
Jeffrey – Lagerwall  NSC 220
Lam – Smalley  NSC 225
Soares – Tulenko  NSC 218
Valentin – Williams, J.  NSC 222
Williams, K. – Zuccala  NSC 228

WEDNESDAY 4/13 IS EXAM MAKE-UP DAY:
NO REGULAR LECTURE
Exam #2
Tuesday 4/12, 9:15 PM - 10:15 PM

No student may leave exam room in first 15 minutes (i.e. prior to 9:30 PM)

No student may enter exam room after first 15 minutes (i.e. 9:30 PM or later)

BRING YOUR UB CARD
channel 1
How many interfaces can a (user-defined) class implement?

A. None
B. Exactly one
C. One or more
D. Zero or more
E. No more than eight \((2^3)\)
How many interfaces can a (user-defined) class implement?

A. None
B. Exactly one
C. One or more
D. Zero or more
E. No more than eight ($2^3$)

Convince your neighbor your answer is correct!
How many interfaces can a (user-defined) class implement?

A. None
B. Exactly one
C. One or more
D. Zero or more
E. No more than eight ($2^3$)
Inheritance
Same **type** implications as for interfaces:

instance of subclass belongs to subclass type and superclass type

inheritance: non-private members of superclass can be accessed via subclass object.
implications of the type hierarchy

- hierarchy of types
  - implementation (class to interface)
  - inheritance (class to class, interface to interface)

- assignment
  - variable of type T can be assigned a value of type S
    - where S and T are the same type
    - where S is a subtype of T

- members
  - where S is a subtype of T, all non-private members of both S and T are accessible on an instance of S

- declared vs. actual type
  - using a variable of type T, only methods defined for T may be called, even if object referred to has additional methods
  - when a method is called using a reference whose type is T, but which refers to an object of type S (S a subtype of T), the definition for S is used. Essence of POLYMORPHISM
When it comes to methods defined in a superclass, a subclass has options:

- inherit
- totally override
- partially override
When it comes to methods defined in a superclass, a subclass has options:

- **inherit**
- provide no definition in subclass,
- accept definition from superclass
When it comes to methods defined in a superclass, a subclass has options:

- totally override
- completely reject definition in superclass
- provide entirely new definition in subclass
When it comes to methods defined in a superclass, a subclass has options:

- partially override
- call superclass method, and
- provide additional code in subclass
(see eclipse)
total (complete) overriding

A subclass provides an entirely new definition for a method which would otherwise have been inherited from the superclass.

partial overriding

A subclass provides a definition for a method which would otherwise have been inherited from the superclass, but calls the superclass version via super.

inheritance

A subclass does not provide an alternate definition for a method defined in the superclass, which is inherited.
object layout in memory
When a class is instantiated, memory is reserved for the whole object, including parts contributed by ancestor classes.

```java
public class A {
    private int _x;
    public A(int x) {
        _x = x;
    }
}

public class B extends A {
    private double _y;
    public B(double y) {
        _y = y;
    }
}
```
When creating an instance of B, we have a choice of which constructor to use.

Using the second constructor we can specify the initial values of the instance variables by passing arguments.

This constructor calls the superclass’s constructor.
public class A {
    private int _x;
    public A(int x) {
        _x = x;
    }
}

public class B extends A {
    private double _y;
    private boolean _z;
    public B() {
        this(1.0, false);
    }
    public B(double d, boolean b) {
        super(5);
        _y = d;
        _z = b;
    }
}

When creating an instance of B, we have a choice of which constructor to use.

Using the first constructor "default" values are provided for the two instance variables; their values are set by the second constructor, which is called from the first (the ‘this(1.0,false)’ call).

The second constructor explicitly calls the superclass constructor with argument 5 (the super(5) call).