University at Buffalo
The State University of New York

cse@buffalo

# CSE115 / CSE503
# Introduction to Computer Science I


Dr. Carl Alphonce

343 Davis Hall

alphonce@buffalo.edu


## Office hours:

Tuesday 10:00 AM – 12:00 PM*

Wednesday 4:00 PM – 5:00 PM

Friday 11:00 AM – 12:00 PM

*OR request appointment via e-mail*

**Tuesday adjustments: 11:00 AM – 1:00 PM on 10/11, 11/1 and 12/6*

ROADMAP

# Last time

- Modeling: exclusive association
- null

# Today

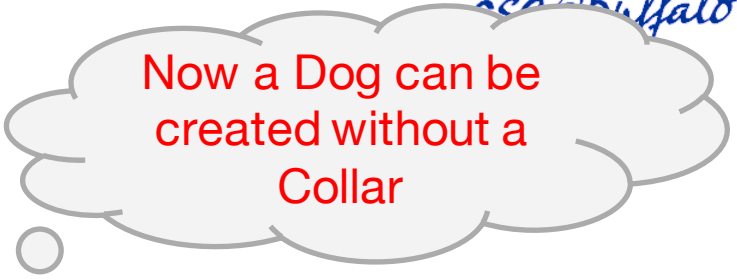
- Interfaces
- Realization relationship

# Coming up

- Graphics
- Event handling

# REVIEW

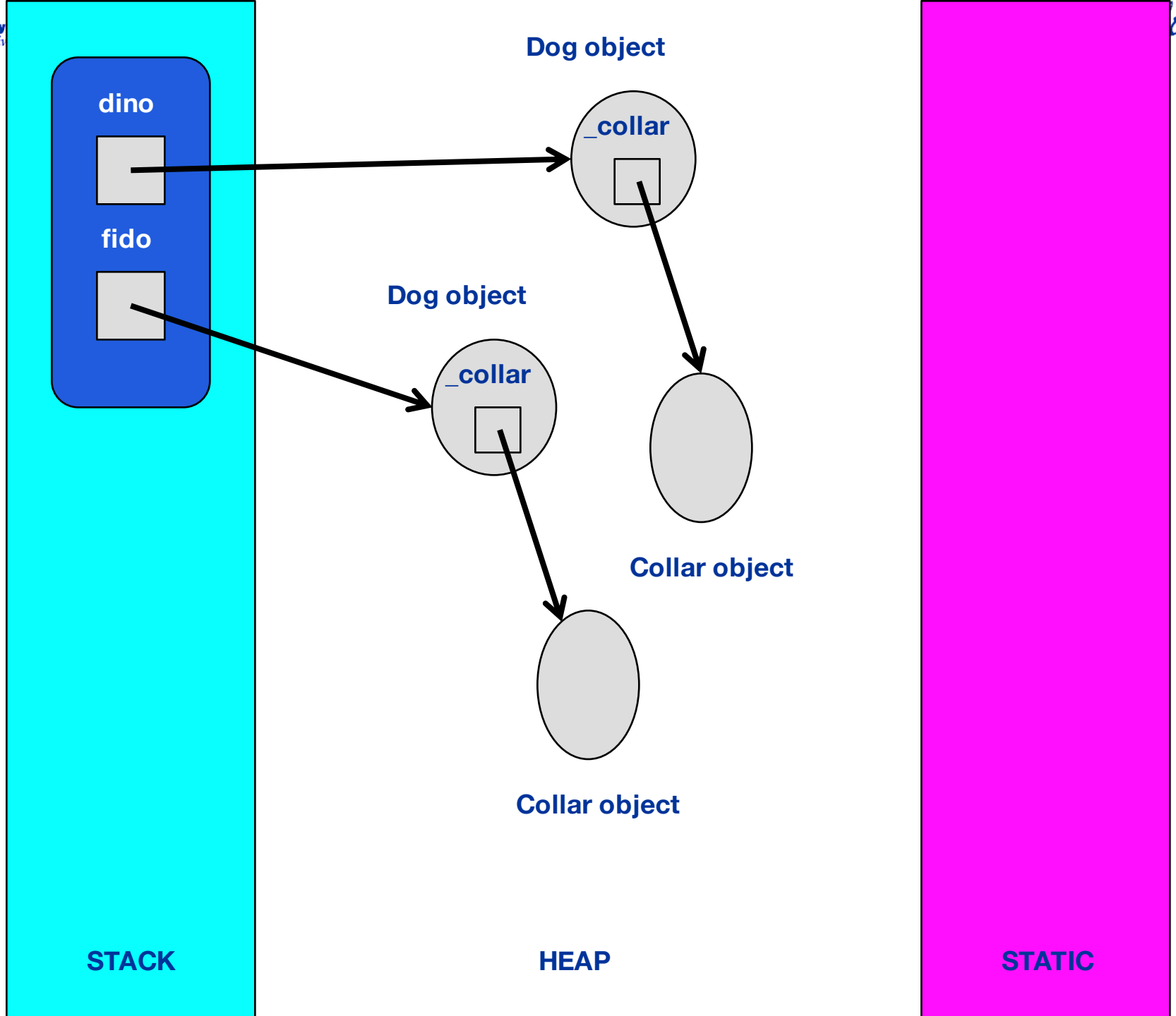
Can also use in constructor




```
public class Dog {
    private Collar _collar;
    public Dog() {
            _collar = null;
    }
    public void setCollar(Collar collar) {
                _collar = collar;
    }
    public Collar removeCollar() {
            Collar temp = _collar;
            _collar = null;
        return temp;
    }
}
```

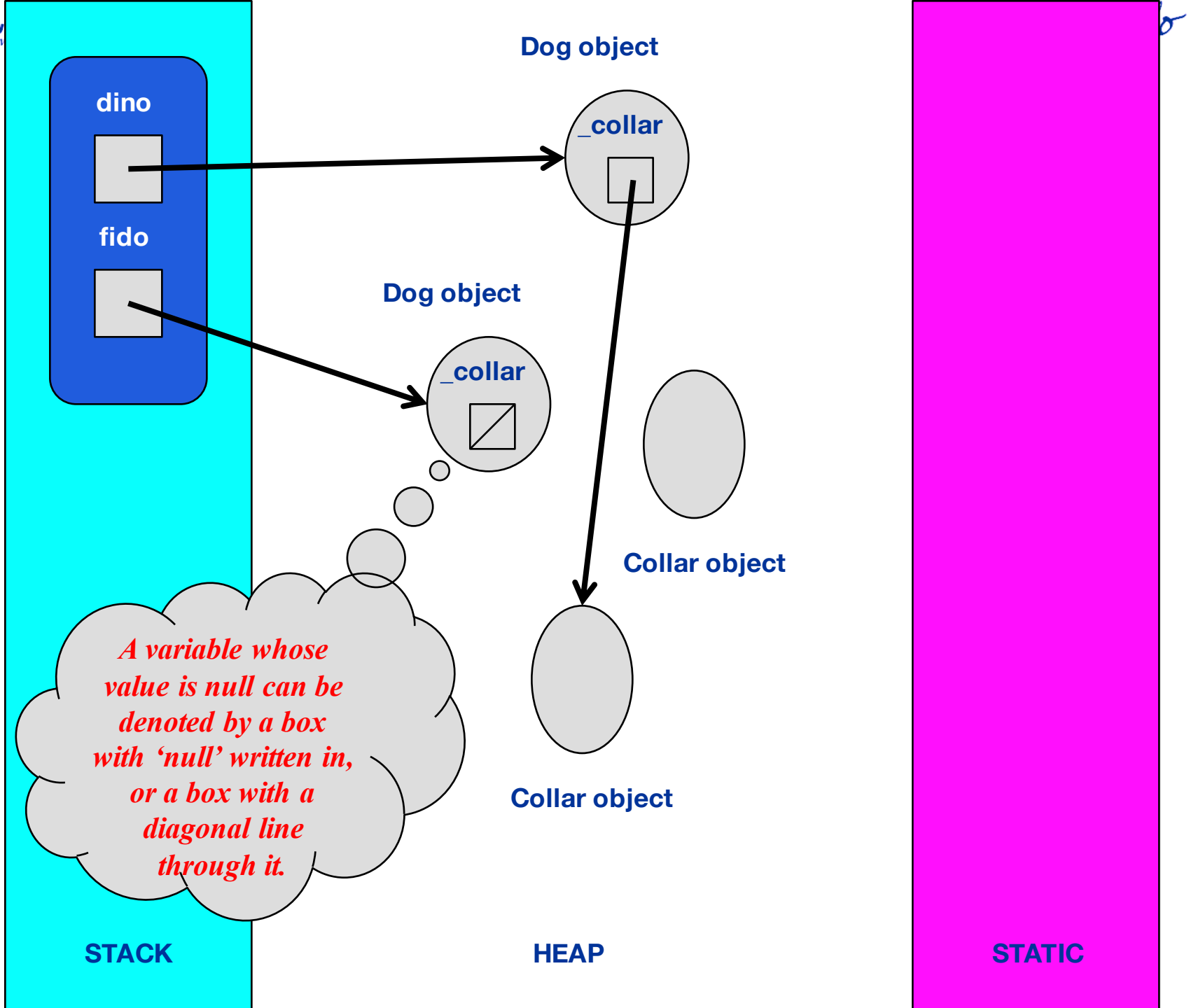
Before call:

`dino.setCollar(fido.removeCollar())`



After call:

`dino.setCollar(fido.removeCollar())`



# ixLearns quiz

# INTERFACES

# POLYMORPHISM

same action, different result

POLYMORPHISM

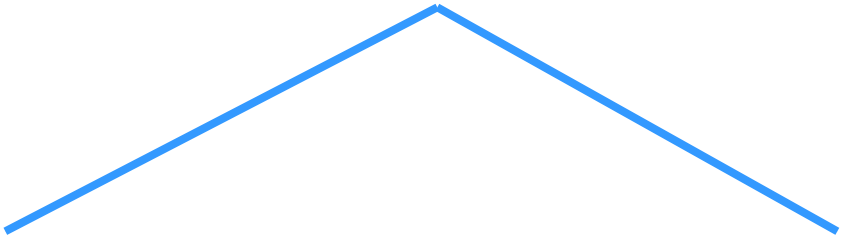
foods

bikes

# PARTICIPATION EXERCISE

# POLYMORPHISM

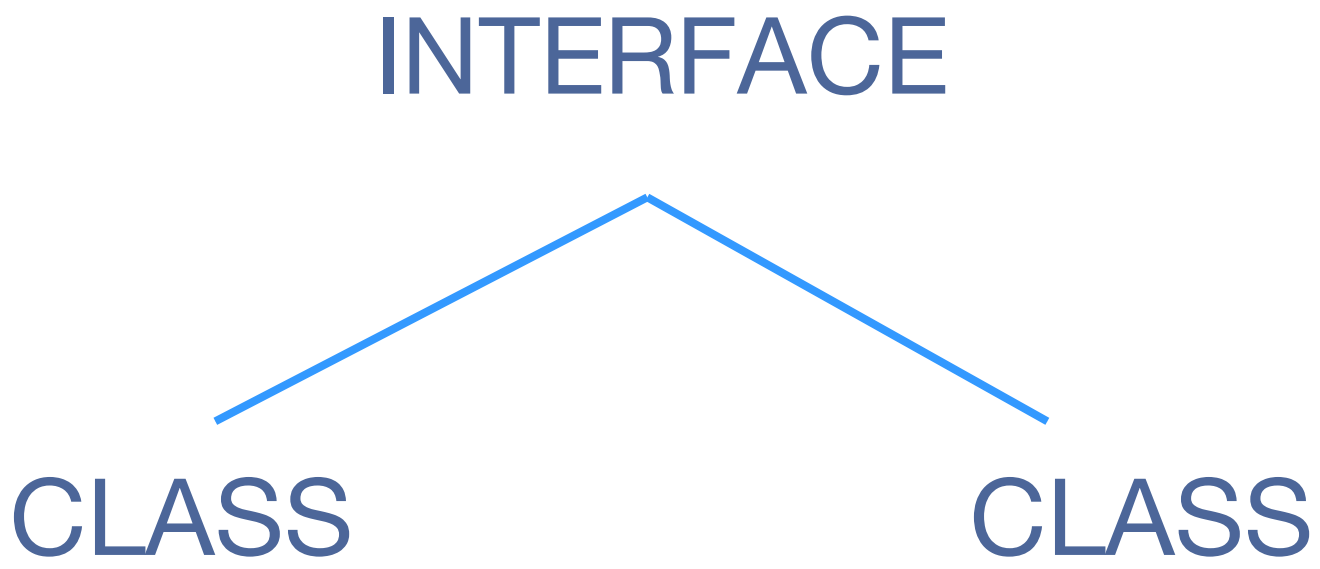
same action



different result

different result

# POLYMORPHISM



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

1) Example from Java's libraries (one detail omitted)

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

2) Example from Java's libraries (one detail omitted)

```
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 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.

# A class can *implement* an interface:

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

# 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

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. CType):

```
public class C1Type implements IType {…}
public class C2Type implements IType {…}

IType var;
var = new C1Type(); // subtype of IType
var = new C2Type(); // subtype of IType
```