CSC 1052 – Algorithms & Data Structures II:

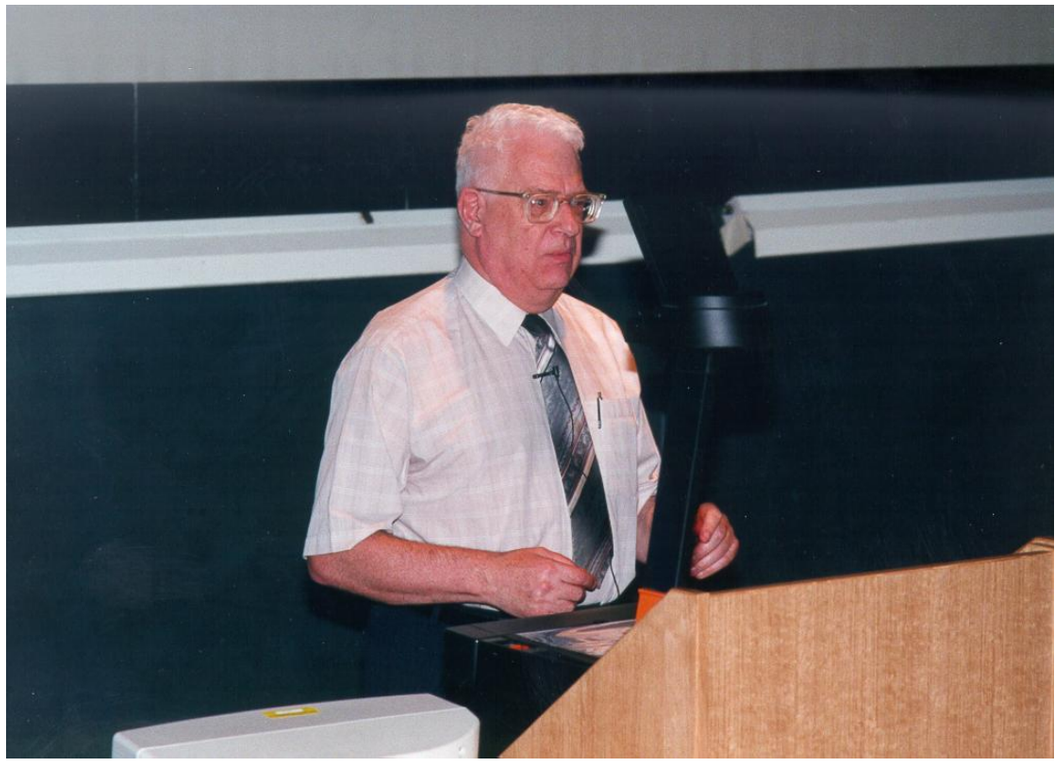
Interfaces

Professor Henry Carter
Spring 2017

Recap

- Evaluating efficiency means many different things depending on context
- Complexity can measure time or space costs
- Order of growth describes how the efficiency decreases as input side grows

Abstraction

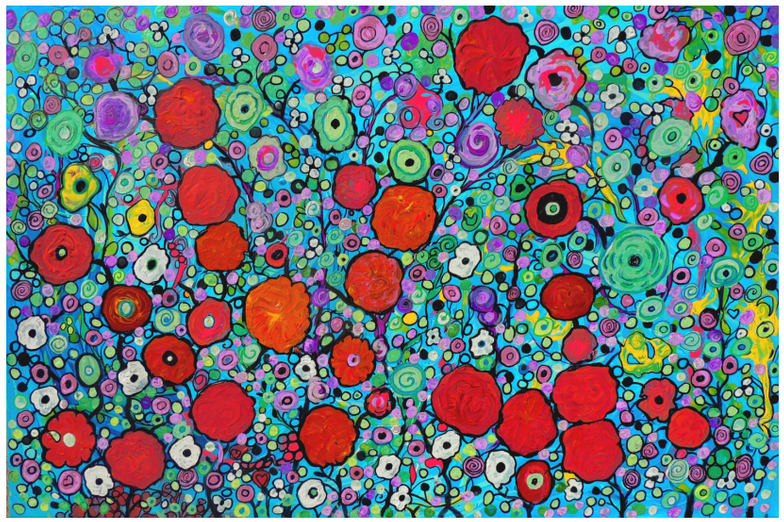


"All problems in computer science can be solved by another level of indirection" – David Wheeler

What is abstraction?

- Hiding details that are not necessary to use an item
- Common inside and outside computing
- Helps users focus on the basic aspects necessary without needing to be an expert

- How do you use abstraction?



Abstraction in programming

- Code in any language is usually divisible into modules
  - What do we call the main modular unit in Java?
- Recall: keywords can be used to limit the availability of certain methods
  - These are details the outside user doesn't need to know
- Higher level languages are, themselves, abstraction
  - How many people write their code as machine instructions?

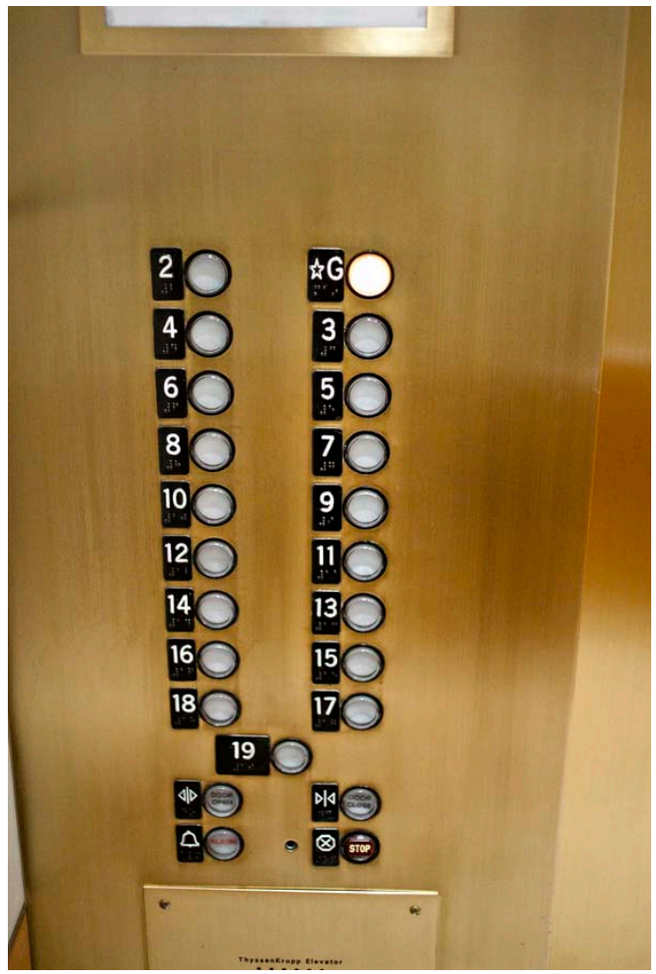
ADTs

- Primitive types provide an encapsulation of simple data
  - Allows creation, observation, and modification
- Classes can represent more complex abstract data
  - The Date class encapsulates a more complex idea than int

- Abstract Data Types refer to any data typed defined by its behavior independent of implementation

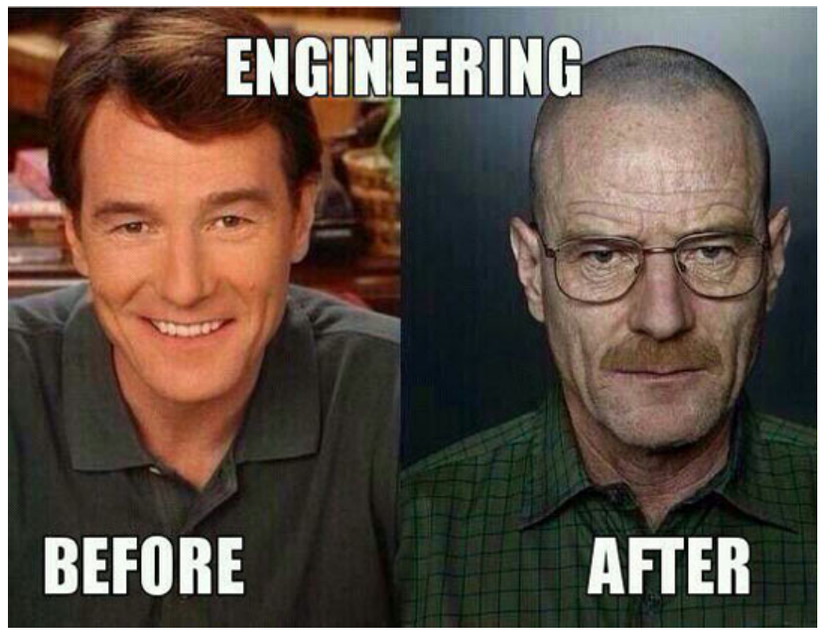
Data levels

- Abstract (logical) level
- Implementation (concrete) level
- Application (user) level



What should ADTs define?

- Core/logical functionality (methods)
- Preconditions
- Postconditions



Java Interface

- Formal definition of an ADT
- Allows the declaration of variables and methods (like a class)
- All variables must be constant and all methods must be abstract

Example Interface

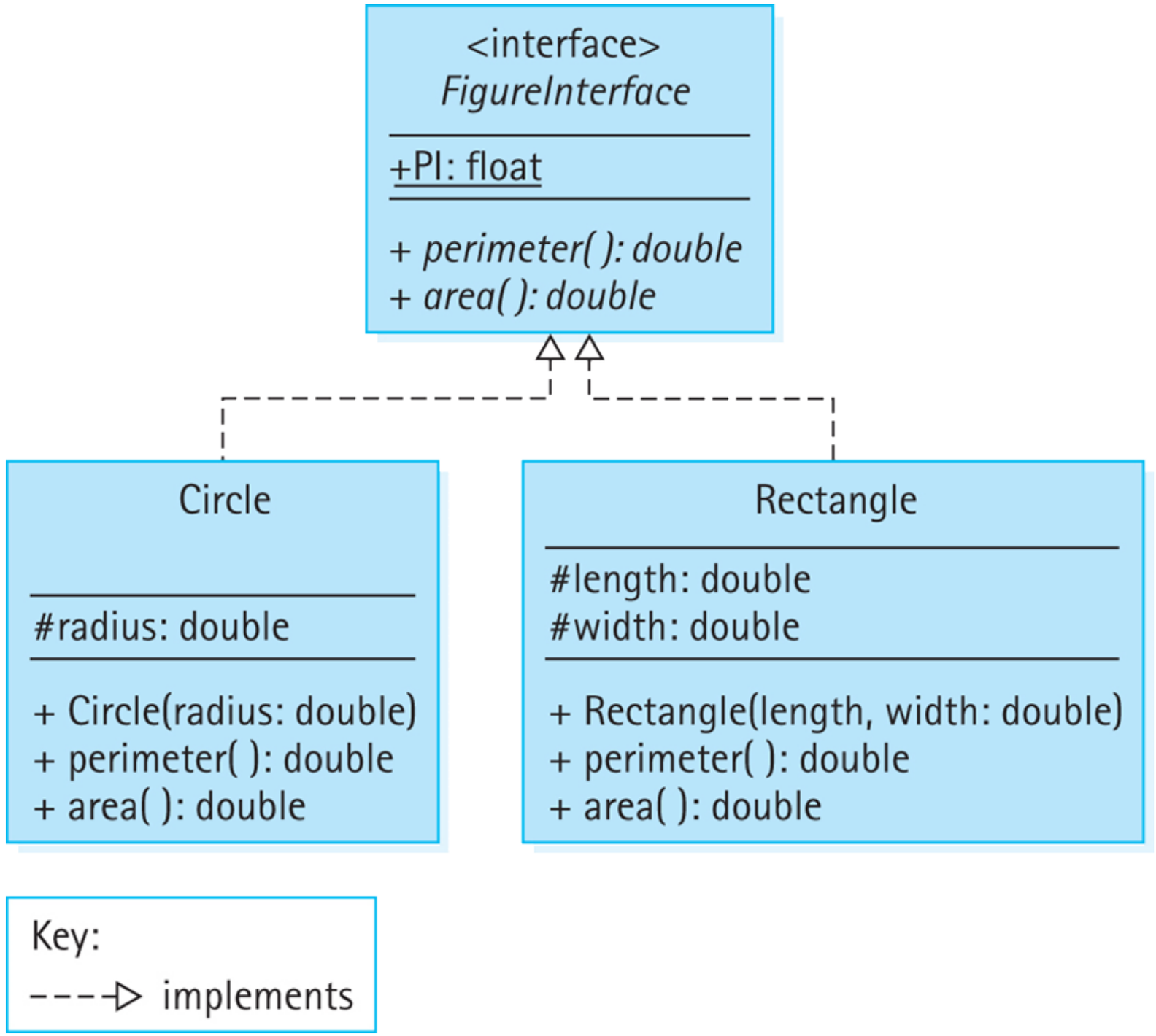
```
package ch02.figures;

public interface FigureInterface
{
    final double PI = 3.14;

    double perimeter();
    // Returns perimeter of this figure.

    double area();
    // Returns area of this figure.
}
```

Example Implementation

```
package ch02.figures;

public class Circle implements
                     FigureInterface
{
  protected double radius;

  public Circle(double radius)
  {
    this.radius = radius;
  }

  public double perimeter()
  {
    return(2 * PI * radius);
  }

  public double area()
  {
    return(PI * radius * radius);
  }
}
```

```
package ch02.figures;

public class Rectangle implements
                          FigureInterface
{
  protected double length, width;

  public Rectangle(double length,
                   double width)
  {
    this.length = length;
    this.width = width;
  }

  public double perimeter()
  {
    return(2 * (length + width));
  }

  public double area()
  {
    return(length * width);
  }
}
```

UML diagram


<interface>
FigureInterface
+PI: float
+ perimeter( ): double
+ area( ): double
Circle
#radius: double
+ Circle(radius: double)
+ perimeter( ): double
+ area( ): double
Rectangle
#length: double
#width: double
+ Rectangle(length, width: double)
+ perimeter( ): double
+ area( ): double
Key:
implements

Why make an interface with no code?

- Template for developers
- Compiler-assisted error checking
- Allows for modular swapping of implementations

Polymorphism strikes again

- Recall: we can store objects of different classes in the same variable
  - Inheritance based polymorphism
- Variables can be declared as an interface
- Objects stored must implement the declared interface

Example Polymorphism

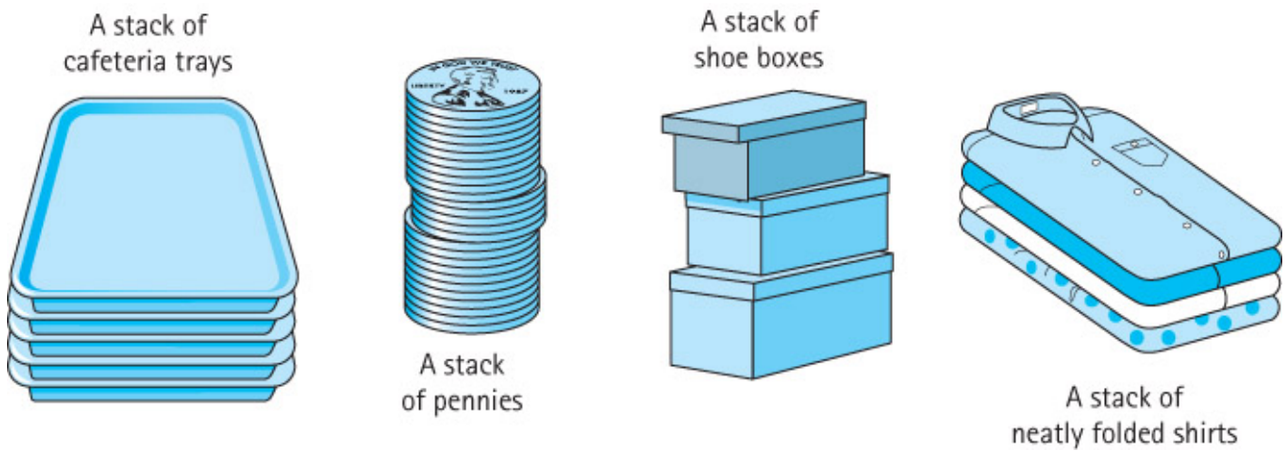
```
FigureInterface[] figures = new FigureInterface[COUNT];
final int COUNT = 5;

// generate figures
for (int i = 0; i < COUNT; i++)
{
  switch (rand.nextInt(2))
  {
    case 0: figures[i] = new Circle(1.0);
    break;

    case 1: figures[i] = new Rectangle(1.0, 2.0);
    break;
  }
}
```

Debug this

```
FigureInterface[] myFigs;


myFigs = new FigureInterface[3];
myFigs[0] = new FigureInterface();
myFigs[1] = new FigureInterface();
myFigs[2] = new Circle(1.5);


System.out.println(myFigs[1].area());
```

Stack: abstract level

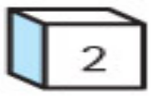
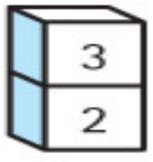
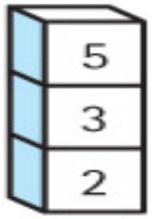
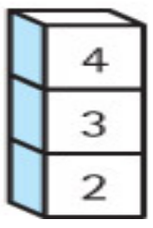
- Modeled after the acclaimed storage method
- Allows fast access to the element on top
- Follows LIFO behavior
  - Order-dependent



Stack operations

- Push
- Pop
- Top (Peek)

Example Operations

originally		stack is empty
push block2		top = block2
push block3		top = block3
push block5		top = block5
pop		top = block3
push block4		top = block4

Stack: what is it good for?

- Saving state
  - Call stack
- Processing nested information
  - Compilers
- Backtracking
  - Depth-first search

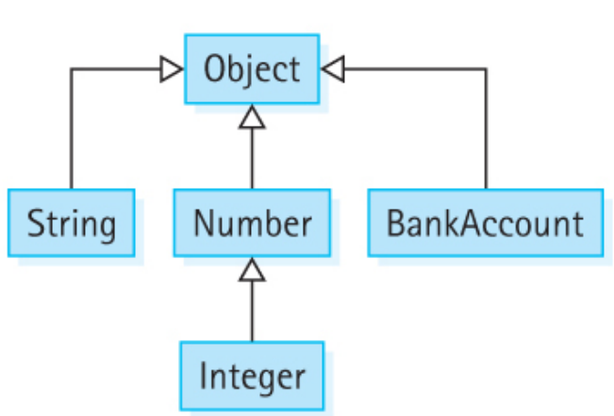
Before we implement…

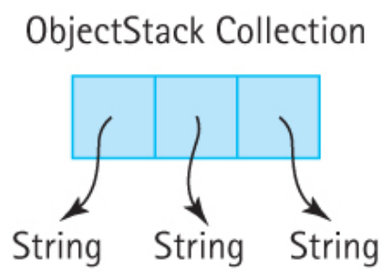
- If we want a stack of integers, we could code this quickly
- If we later want a stack of Strings, we could comb though and refactor
- If we later want…
- We REALLY want an ADT to be able to store many different types of data depending on the application

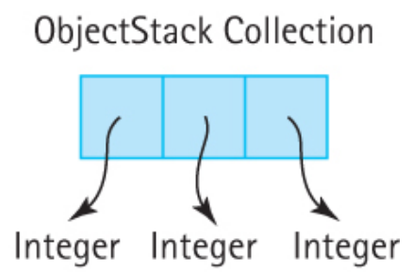
General Purpose Collections

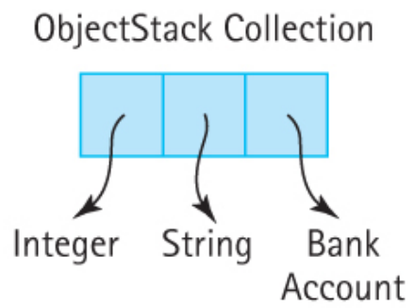
- Collections of class OBJECT
  - Polymorphism
- Collections of an Interface
  - Polymorphism
- Generic Collections
  - Runtime declaration

Object collections

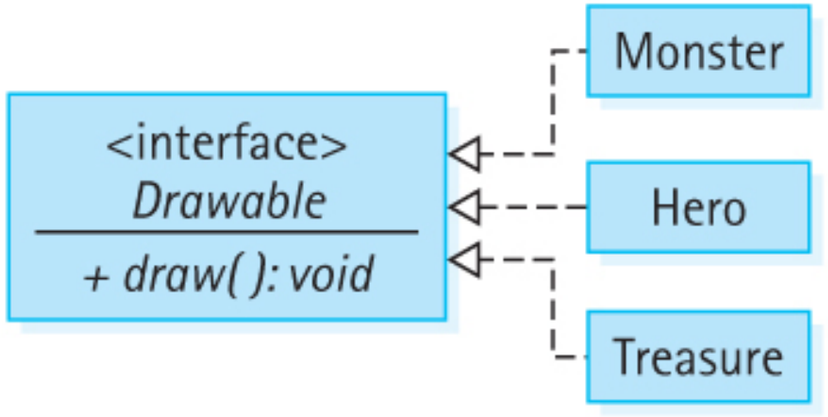


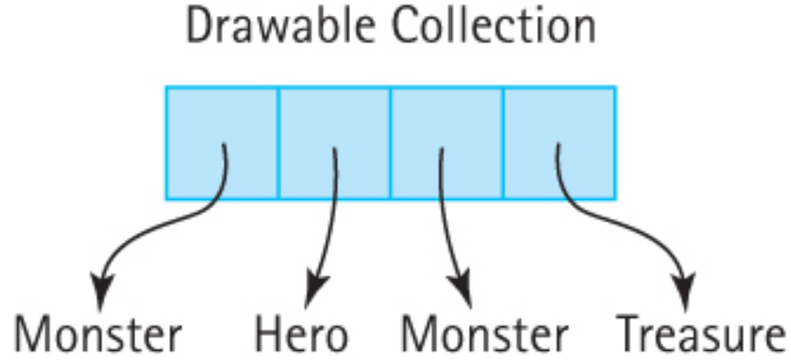




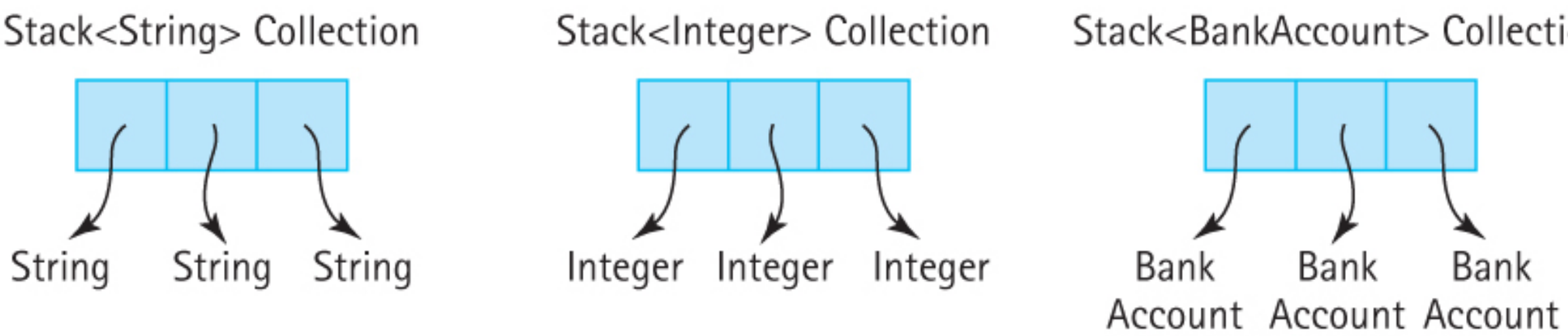


Interface collections





Generic collections



Generic syntax

- Generic class definition:

```
public class Stack<T>
{
    protected T[] elements;
    protected int topIndex = -1;
…
```

- Variable declaration and initialization:

```
Stack<Integer> numbers;
Stack<BankAccount> investments –
            new Stack<BankAccount>();
```

Recap

- Abstraction allows for information to be compartmentalized and simplifies modular use
- Interfaces are the Java construction for formal abstraction
- The stack is a LIFO data structure that allows efficient access to the top element
- Generic collections allow for simplified implementation of data structures that may hold a variety of classes

Next Time...

- Dale, Joyce, Weems Chapter 2.4-2.5
  - Remember, you need to read it BEFORE you come to class!
- Check the course webpage for practice problems
- Peer Tutors
  - http://www.csc.villanova.edu/help/