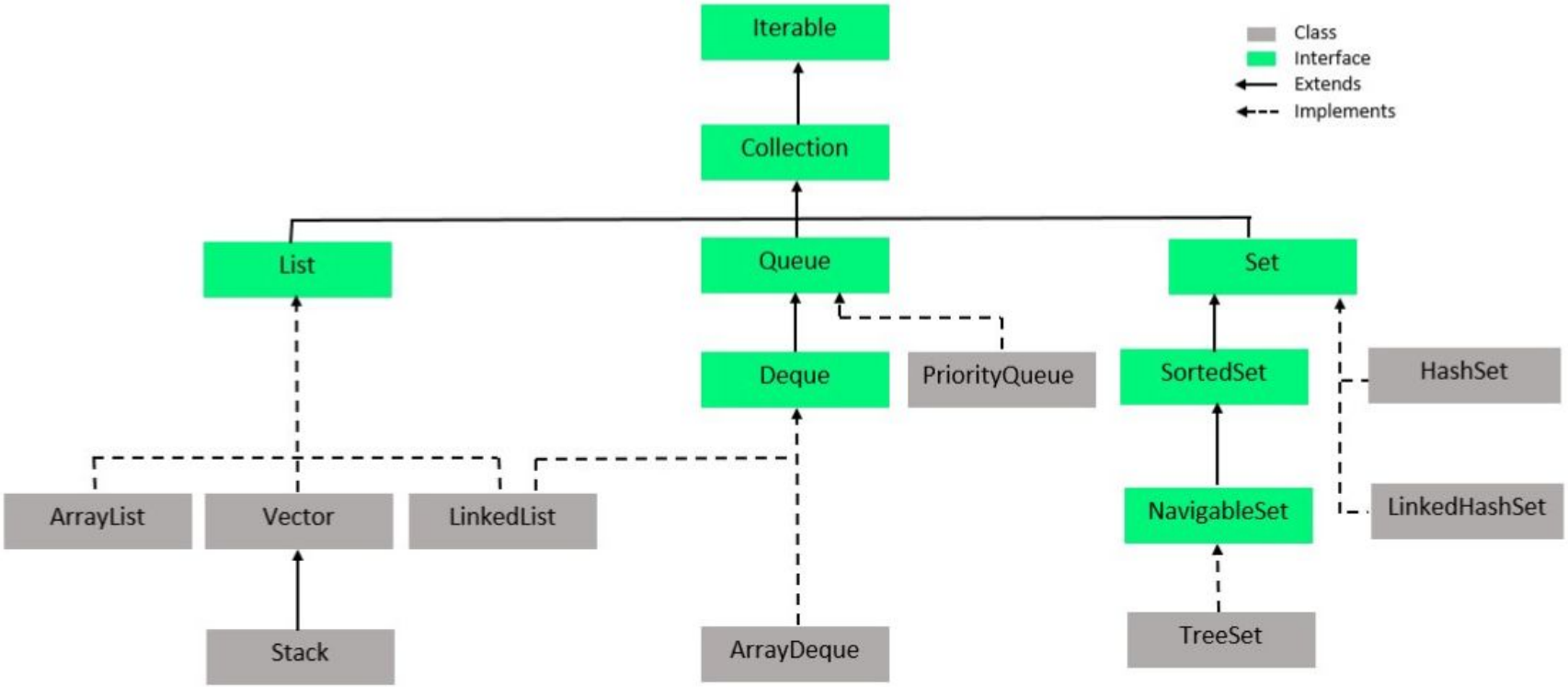
COSC 121: Computer Programming II



Today's Key Concepts

- The Java Collection Framework provides a set of interfaces and classes to support various data operations with common data structures
- Different data structures have different ways of organizing information, optimizing how data is stored, accessed, and manipulated
- Central objective is efficiency, with main tradeoffs between time and space
- Major categories of data structure interfaces:
  - List: ArrayList
  - Others in future classes
- Using an Iterator for Collections
- Java generics that is compatible with various data types
- Techniques used in various data structures and algorithms

Java Collection Framework

- An architecture made up of interfaces and classes
  - Components are ready to use for various programming needs
  - Offers different pre-programmed data operations like searching, sorting, insertion, deletion, and manipulation
  - All of the classes and interfaces here are available through java.util
- Root class: Iterable
  - Has an iterator function that allows the user to traverse through all of the collection class objects
  - Collection (extends Iterable) has basic methods to add, delete, and manipulate data
- Map
  - Separate hierarchy but also part of the framework (next course)

ArrayList Class

- ArrayList implements the List interface
  - Elements in a list are ordered like a sequence
- ArrayList objects are dynamic arrays
  - Meaning, a resizable implementation that dynamically grows as needed
  - Implemented internally as an array, and therefore:
    - Has a capacity that can be extended when needed
    - Has a size that stores the current number of items
    - Data can be accessed via an index (like an array)
    - Supports random access, directly accessing data in constant time
  - Not ideal in its use of memory
  - Has additional methods to support flexible data manipulation (e.g., insert and delete within list)

How might you implement such a class?

Example

```
public class TestArrayList {
 public static void main(String[] args) {
     ArrayList<String> fruits = new ArrayList<>();

     // 2. Add the first four elements using the add(E element) method
     fruits.add("Apple");
     fruits.add("Banana");
     fruits.add("Orange");
     fruits.add("Mango");
     System.out.println( "Original list: " + fruits + " has " + fruits.size() + " elements");
```

Original list: [Apple, Banana, Orange, Mango] has 4 elements

```
 }
}
```

Example

```
public class TestArrayList {
 public static void main(String[] args) {
     ArrayList<String> fruits = new ArrayList<>();

     // 2. Add the first four elements using the add(E element) method
     fruits.add("Apple");
     fruits.add("Banana");
     fruits.add("Orange");
     fruits.add("Mango");
     System.out.println( "Original list: " + fruits + " has " + fruits.size() + " elements");

     fruits.add( 2, "Kiwi" );                                    // 3. Insert element
     System.out.println( fruits );

 }
}
```

Original list: [Apple, Banana, Orange, Mango] has 4 elements

[Apple, Banana, Kiwi, Orange, Mango]

Example

```
public class TestArrayList {
 public static void main(String[] args) {
     ArrayList<String> fruits = new ArrayList<>();

     // 2. Add the first four elements using the add(E element) method
     fruits.add("Apple");
     fruits.add("Banana");
     fruits.add("Orange");
     fruits.add("Mango");
     System.out.println( "Original list: " + fruits + " has " + fruits.size() + " elements");

     fruits.add( 2, "Kiwi" );
     System.out.println( fruits );

     fruits.remove( 1 );
     System.out.println( fruits );

 }
}
```

Original list: [Apple, Banana, Orange, Mango] has 4 elements

// 3. Insert element

[Apple, Banana, Kiwi, Orange, Mango]

// 4. Remove element

[Apple, Kiwi, Orange, Mango]

Example

```
public class TestArrayList {
 public static void main(String[] args) {
     ArrayList<String> fruits = new ArrayList<>();

     // 2. Add the first four elements using the add(E element) method
     fruits.add("Apple");
     fruits.add("Banana");
     fruits.add("Orange");
     fruits.add("Mango");
     System.out.println( "Original list: " + fruits + " has " + fruits.size() + " elements");

     fruits.add( 2, "Kiwi" );                          // 3. Insert element
     System.out.println( fruits );

     fruits.remove( 1 );                               // 4. Remove element
     System.out.println( fruits );

     if(fruits.contains("Papaya"))                     // 5. Check an element
         System.out.println("The list has this fruit already");
     else
         fruits.add("Papaya");
     System.out.println( fruits );



 }
}
```

Original list: [Apple, Banana, Orange, Mango] has 4 elements

[Apple, Banana, Kiwi, Orange, Mango]

[Apple, Kiwi, Orange, Mango]

[Apple, Kiwi, Orange, Mango, Papaya]

Example

```
public class TestArrayList {
 public static void main(String[] args) {
     ArrayList<String> fruits = new ArrayList<>();

     // 2. Add the first four elements using the add(E element) method
     fruits.add("Apple");
     fruits.add("Banana");
     fruits.add("Orange");
     fruits.add("Mango");
     System.out.println( "Original list: " + fruits + " has " + fruits.size() + " elements");

     fruits.add( 2, "Kiwi" );                          // 3. Insert element
     System.out.println( fruits );

     fruits.remove( 1 );                               // 4. Remove element
     System.out.println( fruits );

     if(fruits.contains("Papaya"))                     // 5. Check an element
         System.out.println("The list has this fruit already");
     else
         fruits.add("Papaya");
     System.out.println( fruits );

     // 6. Use get methods
     System.out.println( "The fruit at index 2 is " + fruits.get( 2 ));
     System.out.println( "The last fruit now is " + fruits.getLast());
 }
}
```

Original list: [Apple, Banana, Orange, Mango] has 4 elements

[Apple, Banana, Kiwi, Orange, Mango]

[Apple, Kiwi, Orange, Mango]

[Apple, Kiwi, Orange, Mango, Papaya]

The fruit at index 2 is Orange
The last fruit now is Papaya

Options for Declaring ArrayList Objects

- General Syntax: ArrayList<?> arrayListName = **new** ArrayList<?>();
  - Where ? is replaced by the type

Options for Declaring ArrayList Objects

- General Syntax: ArrayList<?> arrayListName = **new** ArrayList<?>();
  - Where ? is replaced by the type
- Examples with default capacity:
  - ArrayList<String> alist = **new** ArrayList<String>();
  - ArrayList<String> alist = **new** ArrayList<>();
- Example with specific capacity:
  - ArrayList<Date> calendar = **new** ArrayList<Date>(100);

Options for Declaring ArrayList Objects

- General Syntax: ArrayList<?> arrayListName = **new** ArrayList<?>();
  - Where ? is replaced by the type
- Examples with default capacity:
  - ArrayList<String> alist = **new** ArrayList<String>();
  - ArrayList<String> alist = **new** ArrayList<>();
- Example with specific capacity:
  - ArrayList<Date> calendar = **new** ArrayList<Date>(100);
- Example with no type so defaults to Object type:
  - ArrayList alist = **new** ArrayList();

Options for Declaring ArrayList Objects

- General Syntax: ArrayList<?> arrayListName = **new** ArrayList<?>();
  - Where ? is replaced by the type
- Examples with default capacity:
  - ArrayList<String> alist = **new** ArrayList<String>();
  - ArrayList<String> alist = **new** ArrayList<>();
- Example with specific capacity:
  - ArrayList<Date> calendar = **new** ArrayList<Date>(100);
- Example with no type so defaults to Object type:
  - ArrayList alist = **new** ArrayList();
- Example from another list:
  - ArrayList<String> basket = **new** ArrayList<>(fruits);
  - ArrayList<String> present = **new** ArrayList<>(Arrays.*asList*("Banana","Melon"));

iClicker Question

Which of the following is used to construct an ArrayList?

A. new ArrayList[ ]
B. new ArrayList[100]
C. new ArrayList<>()
D. There is more than one correct answer

iClicker Question

Suppose ArrayList x has two strings ["A", "B"]. Which of the following will cause x to become ["A", "C", "B"] ?

A. x[1] = "C";
B. x.add("C");
C. x.add(0, "C");
D. x.add(1, "C");
E. x.add(2, "C");

Some Useful Methods

boolean add(E o)	Appends a new element o at the end of this list.
void add(int index, E o)	Adds a new element o at the specified index in this list.
E set(int index, E e)	Sets the element at the specified index.
E get(int index)	Returns the element from this list at the specified index.
boolean remove(Object e)	Removes the first occurrence of element o from this list.
E remove(int index)	Removes the element at the specified index.
void clear()	Removes all the elements from this list.
int indexOf(Object e)	Returns the index of the first matching element in this list
int lastIndexOf(Object e)	Returns the index of the last matching element in this list.
boolean contains(Object e)	Returns true if this list contains the element o.
boolean isEmpty()	Returns true if this list contains no elements.
int size()	Returns the number of elements in this list.
void trimToSize()	Trims the capacity of this ArrayList instance to the current size.
boolean equals(Object x)	Returns true if **x** is a list and both lists have the same size, and all corresponding pairs of elements are *equal*.
String toString()	Returns a string representation of the list elements.

*clicker question

*clicker question

iClicker Question

Suppose ArrayList x has two strings ["A", "B"]. Which of the following will cause x to become ["A"] ?

A. x[1] = "";
B. x.remove(0);
C. x.remove(1);
D. x.remove("B");
E. There is more than one correct answer

iClicker Question

Suppose ArrayList x has two strings ["A", "B"]. Which of the following will cause x to become ["C"] ?

A.
```
x = null;
x.add( "C" );
```
B.
```
x.clear();
x.add( "C" );
```
C.
```
x.remove( 0 );
x.set( 0, "C" );
```
D.
```
x.remove( "B" );
```

iClicker Question

What is the output of this code?

```
ArrayList<String> x =
        new ArrayList<>(Arrays.asList("A","B","C"));
System.out.println(x);
x.set(1, "D");
x.add("D");
System.out.println(x);
System.out.println(x.indexOf("C"));
System.out.println(x.indexOf("Z"));
System.out.println(x.remove("Z"));
System.out.println(x.remove("D"));
System.out.println(x);
```

A.
```
[A, B, C]
[D, B, C, D]
2
-1
false
true
[B, C, D]
```

B.
```
[A, B, C]
[A, D, C, D]
2
-1
false
true
[A, C, D]
```

C.
```
[A, B, C]
[A, D, C, D]
2
-1
false
true
[A, D, C]
```

D.
```
[A, B, C]
[A, D, C, D]
-1
-1
false
true
[A, C, D]
```

Practice Program

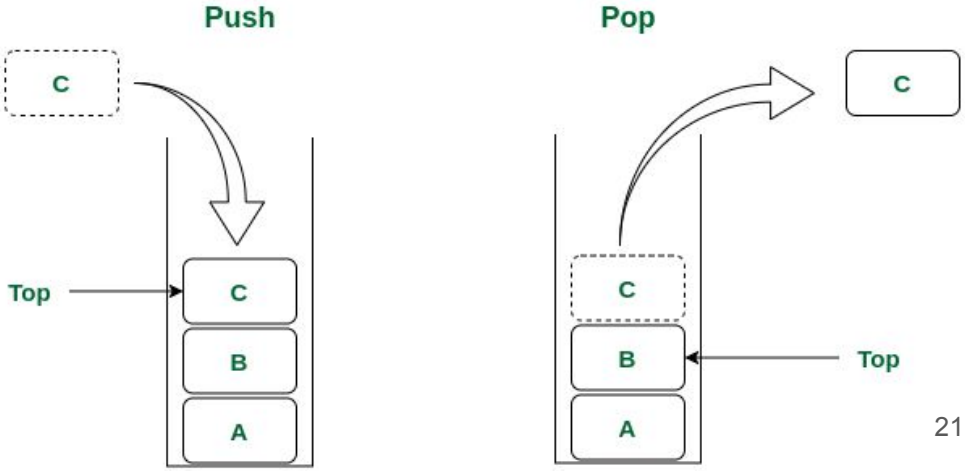
- Write a program that reads integers from the user and stores them into an ArrayList without storing duplicates. The input -1 ends the program.
- Example interaction:

```
Enter positive integers (or -1 to end):
45
32
19
21
32
45
78
11
-1
We stored: [45, 32, 19, 21, 78, 11]
```

- See: TestArrayListUserInput.java

Implementing Stack

- A stack is a first-in-last-out data structure
- Common operations:
  - isEmpty(): boolean
  - getSize(): int
  - peek(): Object (returns top element without removing it)
  - pop(): Object
  - push(o Object): void
- Implement a stack using an ArrayList
- See TestStack.java



Using Iterators

- You want to apply the same operation to a list of robots
- Use an iterator!
  - No need to know the length of the array
  - No need to specify increment
- Example:

```
ArrayList<Robot> robots = new ArrayList<>();
for ( int i = 0; i < NUM_ROBOTS; i++ )
        robots.add(new Robot( … ) );          // don't worry about constructor details
for ( Robot robot : robots )
        robot.walk();
```

- See Robot.java and RobotAnimation.java

Last Example

- You may also shoot bullets at robots which needs to be tracked as they move but they need to be removed once they pass the screen
- Here, you need a separate Iterator to loop through the bullets because you can't change the list while the code is iterating through it
- Example:

```
ArrayList<Bullet> bullets = new ArrayList<>();
Iterator<Bullet> bulletIter = bullets.iterator();
while( bulletIter.hasNext() ) {
        Bullet b = bulletIter.next();
        b.y -= BULLET_SPEED;
        if (b.y < 0) bulletIter.remove();
}
```

- See AsteroidsGame.java