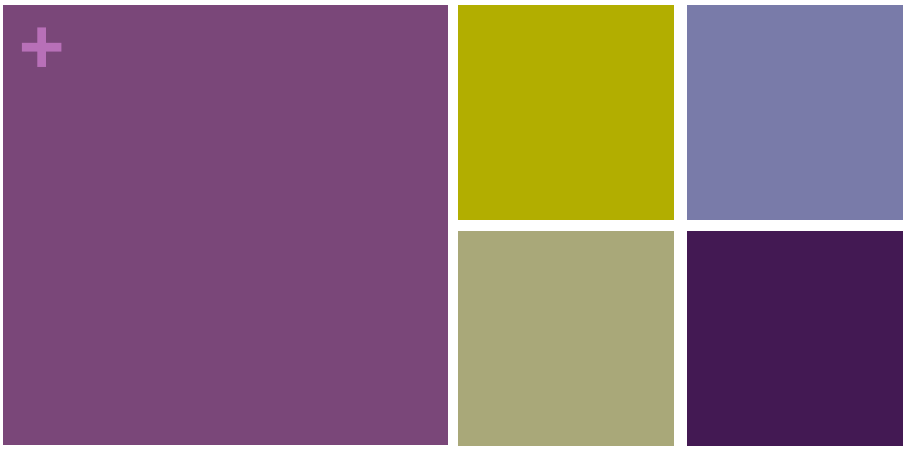

ADTs, Collections, Implementing ArrayList

The Collection Interface

```
public interface Collection {
    public int size();
    public boolean isEmpty();
    public boolean contains(Object o);
    public Iterator iterator();
    public Object[] toArray();
    public Object[] toArray(Object[] a);
    public boolean add(Object e);
    public boolean remove(Object o);
    public boolean containsAll(Collection c);
    public boolean addAll(Collection c);
    public boolean removeAll(Collection c);
    public boolean retainAll(Collection c);
    public void clear();
}
```

The Collection Interface <Generic>

```
public interface Collection<E> {
    public int size();
    public boolean isEmpty();
    public boolean contains(E o);
    public Iterator iterator();
    public E[] toArray();
    public E[] toArray(E[] a);
    public boolean add(E e);
    public boolean remove(E o);
    public boolean containsAll(Collection<E> c);
    public boolean addAll(Collection<E> c);
    public boolean removeAll(Collection<E> c);
    public boolean retainAll(Collection<E> c);
    public void clear();
}
```

The List Interface

```
public interface List extends Collection {
    // everything from Collection
    public Object get(int index);
    public Object set(int index, Object element);
    public void add(int index, Object element);
    public Object remove(int index);
    public int indexOf(Object o);
    public int lastIndexOf(Object o);
    public ListIterator listIterator();
    public ListIterator listIterator(int index);
    public List subList(int fromIndex, int toIndex);
}
```

The List Interface <Generic>

```
public interface List<E> extends Collection<E> {
    // everything from Collection
    public E get(int index);
    public E set(int index, E element);
    public void add(int index, E element);
    public E remove(int index);
    public int indexOf(E o);
    public int lastIndexOf(E o);
    public ListIterator<E> listIterator();
    public ListIterator<E> listIterator(int index);
    public List<E> subList(int fromIndex, int toIndex);

}
```

ArrayList<E> implements the list interface

```
public interface List<E> extends Collection<E> {
    // everything from Collection
    public E get(int index);
    public E set(int index, E element);
    public void add(int index, E element);
    public E remove(int index);
    public int indexOf(E o);
    public int lastIndexOf(E o);
    public ListIterator<E> listIterator();
    public ListIterator<E> listIterator(int index);
    public List<E> subList(int fromIndex, int toIndex);

}
```

Using ArrayLists

- **ArrayList<Place> places;**

  places = new ArrayList<Place>();

  places.add(new Place("19010", "Bryn Mawr", "PA");
  places.add(new Place("19146", "Philadelphia", "PA");
  places.add(new Place("87501", "Santa Fe", "NM");

- Place brynMawr = places.get(?); // what index do we need to get Bryn Mawr?
- What do I get at index 3? How about index 2?

ArrayList implementation

- Version One: Fixed array
- Version Two: Making it Generic
- Version Three: Dynamic Array

Simplified List Interface (ADT)

```
public interface SimplifiedList {
        public boolean add(Object item);
        public boolean add(int index, Object item);

        public Object remove(int index);
        public Object set(int index, Object item);
        public Object get(int index);

        public boolean contains (Object item);

        public boolean isEmpty();
        public void clear();
        public int size();

        public boolean isFull();
 } // interface SimplifiedList
```

Implementing SimplifiedList

- **Data Structure#1: fixed size array (capacity)**
  - **Use an array of Object to store stuff**
  - **A variable to store # entries in it**
  - **Its capacity (max size)**

Version 2: Making it Generic

- Change the interface and the class

Simplified List Interface (ADT) <Generic>

```
public interface SimplifiedList<E> {
        public boolean add(E item);
        public boolean add(int index, E item);

        public E remove(int index);
        public E set(int index, E item);
        public E get(int index);

        public boolean contains (E item);

        public boolean isEmpty();
        public void clear();
        public int size();

        public boolean isFull();
 } // interface SimplifiedList
```

Version 3: Using flexible arrays

- add a method called void reallocate() that
  - is called when adding to an array that is at capacity
  - doubles the size of the array copying the current values to the new values.

```
private void reallocate() {
   capacity = 2 * capacity;
   theData = Arrays.copyOf(theData, capacity);
}
```

Exam 1 concepts

- Order of operations for arithmetic, assignment, and logical expressions
- Widening vs. Narrowing operations.
- integer vs. floating point arithmetic
- The String class and the methods we've used with it
- .equals() vs. ==
- How to write a java program to do basic input and output
- the toString() method and how to override it
- Exceptions and try/catch blocks in Java
- primitive vs. object variables
- escape characters
- using String[] split()
- arrays and for loops to use them
- ability to diagram memory allocation. (primitive types vs. Object types)
- basic data type initialization vs. object initialization.
- read a simple Java program and write the expected output of the program.
- understand the parts of a class by name/key word.
- understand encapsulation
- write class methods
- write a class from a text definition/description
- understand basic inheritence