CSM 61B Spring 2017

Abstract Classes & Interfaces

Week 5: February 13, 2017

1 An Appealing Appetizer

1.1
```
public interface Consumable {
    public void consume();
}
public abstract class Food implements Consumable {
    String name;
    public abstract void prepare();
    public void play() {
        System.out.println("Mom says, 'Don't play with your food.'");
    }
}
public class Snack extends Food {
    public void prepare() {
        System.out.println("Taking " + name + " out of wrapper");
    }
    public void consume() {
        System.out.println("Snacking on " + name);
    }
}
```

(a) Compare and contrast interfaces and abstract classes.

(b) Do we need the `play` method in `Snack`?

(c) `Consumable chips = new Snack();`
Does this compile?

2 Iterator Interface

In Java, an **iterator** is an object which allows us to traverse a data structure in linear fashion. Every iterator has two methods: `hasNext` and `next`.

```
interface Iterator<E> {
    boolean hasNext();
    E next();
}
```

2.1 Consider the following code that demonstrates the `ArrayIterator`.

```
Integer[] arr = {1, 2, 3, 4, 5, 6};
Iterator<Integer> iter = new ArrayIterator<>(arr);

System.out.println(iter.hasNext());  // true
System.out.println(iter.next());     // 1

System.out.println(iter.next() + 3); // 5

while (iter.hasNext()) {
    System.out.println(iter.next()); // 3 4 5 6
}
```

Define an `ArrayIterator` class that works as described above.

2.2 Define an `IntListIterator` class that adheres to the `Iterator` interface.

2.3 Define a method, `printAll`, that prints every element in an iterator regardless of how the iterator is implemented.

3 Duplicate Iterator *Extra for Experts*

3.1 Fill out `DuplicateIterator` so that it works as used in the example main method. When given two sorted input iterators, the `DuplicateIterator` returns the elements that are within both iterators. Assume that `findNextElement` is correctly implemented.

```
public class DuplicateIterator<T extends Comparable<? super T>> implements Iterator<T> {

    /** Sets the nextElement instance variable to the next duplicate element
      * (or null if there is no remaining duplicate element). */
    private void findNextElement() { ... }

    public static void main(String[] args) {
        Iterator<Integer> iter1 = Arrays.asList(1, 2, 4, 5, 6, 9).iterator();
        Iterator<Integer> iter2 = Arrays.asList(1, 2, 3, 5, 7, 10).iterator();
        DuplicateIterator<Integer> di = new DuplicateIterator<>(iter1, iter2);
        di.forEachRemaining(o -> System.out.print(o.toString() + " ")); // 1 2 5
    }
}
```