CS 61B Discussion 5: Inheritance II Fall 2015

1 Reduce

We'd like to write a method `reduce`, which uses a binary function to accumulate the values of a `List` of integers into a single value. `reduce` will need to take in an object that can operate (through a method) on two integer arguments and return a single integer. Note that `reduce` must work with a range of binary functions (addition and multiplication, for example). Fill in `reduce` and `main`, and define types for `add` and `mult` in the space provided.

```
import java.util.ArrayList;
import java.util.List;
public class ListUtils {
    /** Apply a function of two arguments cumulatively to the
    *    elements of list and return a single accumulated value. */
    static int reduce( ___________ func, List<Integer> list) {








    }
    public static void main(String[] args) {
        ArrayList<Integer> integers = new ArrayList<>();
        integers.add(2); integers.add(3); integers.add(4);
        ___________ add = ____________________;
        ___________ mult = ____________________;
        reduce(add, integers); //Should evaluate to 9
        reduce(mult, integers); //Should evaluate to 24
    }
}

//Add additional classes and interfaces below:
```

2 Exception Handling

Below is an implementation of a `Farm` class. Its only field is an `ArrayList` of animals, and it has three methods: `getAnimal`, `addAnimal`, and `animalCount`. `getAnimal` takes an integer `i` as an argument and returns the i^{th} element of the farm's list of animals.

This implementation produces an `IndexOutOfBoundsException` when we try to get an animal at an index outside of the bounds of our internal `ArrayList`. This could be confusing to a user with no knowledge of our implementation of the `Farm` class. Instead, rewrite the `getAnimal` method so that it catches `IndexOutOfBoundsExceptions` and throws a more descriptive `IllegalArgumentException`.

```java
import java.util.ArrayList;
public class Farm{
    private ArrayList<Animal> animals = new ArrayList<>();

    /** Adds an animal toAdd to the farm. */
    void addAnimal(Animal toAdd){
        animals.add(toAdd);
    }

    /** Takes an index between 0 and animalCount() - 1 (inclusive)
     *  and returns the animal at that index. */
    Animal getAnimal(int index){
        return animals.get(index);
    }

    /** Returns the number of animals on the farm. */
    int animalCount(){
        return animals.size();
    }
}
```

```java
Animal getAnimal(int index){











}
```

3 Comparator

We'd like to sort an `ArrayList` of animals into ascending order, by age. We can accomplish this using `Collections.sort(List<T> list, Comparator<? super T> c)`. Because instances of the `Animal` class (reproduced below) have no natural ordering, `sort` requires that we write an implementation of the `Comparator` interface that can provide an ordering for us.

Note that an implementation of `Comparator` only needs to support pairwise comparison (see the `compare` method). Remember that we would like to sort in ascending order of age, so an `Animal` that is 3 years old should be considered "less than" one that is 5 years old.

```
public interface Comparator<T> {
    /** Compares its two arguments for order.
     *  Returns a negative integer, zero, or a positive integer if the first
     *  argument is less than, equal to, or greater than the second. */
    int compare(T o1, T o2);

    /** Indicates whether some other object is "equal to" this
    *   comparator. */
    boolean equals(Object obj);
}
```

```
import java.util.ArrayList;
import java.util.Collections;
public class Animal {
    protected String name, noise;
    protected int age;
    public Animal(String name, int age) {
        this.name = name;
        this.age = age;
        this.noise = "Huh?";
    }
    /** Returns this animal's age. */
    public int getAge() {
        return this.age;
    }
    public static void main(String[] args) {
        ArrayList<Animal> animals = new ArrayList<>();
        animals.add(new Cat("Garfield", 4));
        animals.add(new Dog("Biscuit", 2));
        _____________________________________; //Initialize comparator
        Collections.sort(animals, __________ );
    }
}
```

```
import java.util.Comparator;
public class AnimalComparator implements Comparator< ________ > {











}
```