

Handed out: Thursday, February 21, 2008

Due: Wednesday February 27, 2008, by 10PM using Web Turnin

In this assignment you will implement different types of ICritters and implement a place for them to co-exist and get to know each other. You will be utilizing polymorphism and keywords to manage how ICritters interact. You will also add an abstract class or two.

KeywordCollection:

Undoubtedly you remember that ICritters are intended to traverse the Interwebs, searching out interesting tidbits for their owners. How to model this? We will use keywords (aka tags). In particular you will implement a keyword system for the ICritters to use to see if they have similar interests with each other. This will be a new class called KeywordCollection. It will represent a set of String keywords. Each ICritter will have an instance variable called "interests" which will be a KeywordCollection. KeywordCollection needs the following public methods:

```

Set<String>    listKeywords()
boolean        addKeyword(String keyword)
boolean        containsKeyword(String keyword)
boolean        removeKeyword(String keyword)
               // All the boolean methods return true if the
               // keyword exists in the collection at the
               // time of the call, and false if it doesn't.
double         correlation(KeywordCollection other)
               // return the percentage of keywords that
               // match between 'this' KeywordCollection and
               // the 'other' KeywordCollection passed as an
               // argument.

```

Here's an example of the correlation method:

```

// Not quite legal java
KeywordCollection kc1 = { "anime", "nintendo", "music", "RPG" };
KeywordCollection kc2 = { "rpg", "anime", "comedy" };

```

kc1.correlation(kc2) ---> returns 0.5 (50%)

There are a total of 4 items in kc1, and two of them ("rpg", and "anime") match contents of kc2, so $2/4 = 0.5$ (50%)

kc2.correlation(kc1) ---> returns 0.6666666666666666 (~67%)

There are a total of 3 items in kc2, and two of them ("rpg", and "anime") match contents of kc1, so $2/3 = 0.66666...$ (67%)

Note that the comparison ignores case (i.e. "rpg" matches keyword "RPG". See the String API).

ICritter goes abstract:

Reimplement the ICritter class to be abstract. Add a new abstract method:

```
void          interact(ICritter other)
```

The interact method will be called on another critter to figure out how "this" critter deals with the "other" critter (passed in as the argument). Note that this method will not alter the "other" critter, only "this" critter.

Also add the following public concrete methods, they map directly to the similarly named methods in KeywordCollection section. So, the listInterests implementation would call interests.listKeywords().

```
Set<String>   listInterests()
boolean       addInterest(String keyword)
boolean       containsInterest(String keyword)
boolean       removeInterest(String keyword)
double        interestCorrelation(ICritter other)
```

ICritterWorld:

Add a new class to icritters.core called ICritterWorld. It will contain a collection of ICritters and provide the following public methods:

```
List<ICritter> listICritters()

boolean        addICritter(ICritter toAdd)
               // returns true if toAdd is already in list,
               // otherwise false

boolean        removeICritter(ICritter toRemove)
               // returns false if toRemove isn't found,
               // otherwise true.

void           runJamboree()
               // iterate through all of the ICritters (in an
               // unspecified order), having each ICritter call
               // the interact method on every other ICritter.
```

ICritters get friendly with each other (or not):

You should add another instance variable (called "friends") to ICritters which will contain a collection of friend ICritters. Add public methods:

```
List<ICritter> getFriends()
boolean        addFriend(ICritter toAdd),
boolean        removeFriend(ICritter toRemove)
               // The two boolean methods return true if the
               // ICritter exists in the friends List at the
               // time of the call, and false if it doesn't.
```

You will be using at least the addFriend method in your interact method.

ICritter sub-classes:

Implement the following hierarchy: LandICritter and MarineICritter will both be abstract sub-classes of ICritter. ICritterDog and ICritterCat will both be concrete sub-classes of LandICritter. ICritterPenguin will be a concrete sub-class of MarineICritter (even though they spend half their lives on land).

Implement the interact methods such that LandICritters always add other LandICritters as friends, but never add MarineICritters. You will have to check the type of the ICritter that the interact method is passed to make this work. (See <http://tinyurl.com/2x3c82> if this seems just wrong).

We will generalize MarineICritters as curious creatures. They will check to see if their interests and the interests of a possible friend correlate at or above 50%, if so they add them as friends.

If you are confused and/or concerned about an ICritter "Zoe" becoming friends with an ICritter "Frederico", but "Frederico" not becoming friends with "Zoe", think of it as that "Zoe" may go to "Frederico" for interesting information, but "Frederico" may not be interested in "Zoe" for information.

ICrittersCaller:

ICrittersCaller will again contain your main method and put your new code through its paces. This round it should: Create two of each of the concrete classes of ICritters, assign them the following names and keywords:

ICritter Type	Name	Keywords
ICritterDog	Spot	Anime, Nintendo, Music, RPG, Games
ICritterDog	Rover	Books, Buttons, Comedy, Comics, Manga
ICritterCat	Felix	Garlic, Nintendo, Buttons, Statistics, Science
ICritterCat	Socks	Canaries, Dogs, Music, Games, Books, Vampire
ICritterPenguin	Zbignew	Games, Music, Anime, Canaries
ICritterPenguin	Jermaine	Science, Buttons, Garlic, Games, Manga

Create an ICritterWorld, add all your ICritters and throw your JVM's first ever ICritterJamboree (yay!).

You should then print out the list of all your ICritters in the following format:

NOTE: that you aren't required to implement the line-breaking used in the Friends list below.

```
ICritterDog "Spot":
  Interests = { Music, Anime, Games, RPG, Nintendo }
  Friends    = { ICritterDog "Rover", ICritterCat "Felix",
                ICritterCat "Socks" }

ICritterDog "Rover":
  Interests = { Manga, Buttons, Books, Comics, Comedy }
  Friends    = { ICritterDog "Spot", ICritterCat "Felix",
                ICritterCat "Socks" }

ICritterCat "Felix":
  Interests = { Buttons, Statistics, Nintendo, Garlic, Science }
  Friends    = { ICritterDog "Spot", ICritterDog "Rover",
                ICritterCat "Socks" }

ICritterCat "Socks":
  Interests = { Canaries, Vampire, Books, Music, Dogs, Games }
  Friends    = { ICritterDog "Spot", ICritterDog "Rover",
                ICritterCat "Felix" }

ICritterPenguin "Zbignew":
  Interests = { Canaries, Anime, Music, Games }
  Friends    = { ICritterDog "Spot", ICritterCat "Socks" }

ICritterPenguin "Jermaine":
  Interests = { Manga, Buttons, Games, Garlic, Science }
  Friends    = { ICritterCat "Felix" }
```

Administrative Stuff:

This is project 5, your project name and turnin folder should reflect this.