CS100J Spring 2006 Assignment A2
Due (submitted on the CMS) on Thursday, 16 February
Monitoring Rhinos

Endangered species

Website http://msnbc.msn.com/id/10877076/ says that, "the rhinoceros population has declined by 90 percent since 1970, with five species remaining in the world today, all of which are endangered. The white and black rhinos are the only species left in Africa... . " Lately, Kenya (with less than 500 rhinos left!) relocated 33 rhinos to Meru National Park, where they hope to protect them from poachers. Poachers kill the rhinos for their horns.

Rhinos are not the only endangered species. Web page http://www.redlist.org/ will tell you that the number of endangered vertebrates (mammals, birds, reptiles, amphibians, and fishes) grew from 3,314 in 1996/98 to 5,188 in 2004. Of the 22,733 evaluated species 23% were endangered. See http://www.worldwildlife.org/endangered for more info on endangered species.

When an animal population is small, the animals can be monitored. Sometimes they will be captured and tagged. Some tags emit a signal, so that one can track the animal. Even in populated places, animals are tagged. Here in Ithaca, one can see deer with tags on their ears wandering in the fields. Gries sees them often in his back yard near Community Corners. Some of the rhino pictures were downloaded from www.birdingafrica.net.

This assignment: monitoring rhinos

This assignment illustrates how Java's classes and objects can be used to maintain data about a collection of things —like individual rhinos in a park. Read the WHOLE handout before you begin to do the assignment. You may do this assignment with one other person. If you are going to work together, then, as soon as possible, get on the CMS for the course and do what is required to form a group.

Requirements

For this assignment, you are required to design and implement two classes. Class `Rhino` is used by to keep track of rhinos. It has lots of fields and methods, but each method is simple. If you do one thing at a time, and start early, you should have little trouble with this assignment. Class `RhinoTester`, a JUnit class, is used to test class `Rhino`. Do not think too much about this class when first reading this handout. Wait until we tell you how to write such classes before starting.

HELP

If you don't know where to start, if you don't understand testing, if you are lost, **SEE SOMEONE IMMEDIATELY**. Gries, a TA, a consultant. Do not wait. Almost 100 of you have never programmed before, and it is reasonable to expect that you may not fully graps everything. But a little one-on-one help can do wonders.

Class Rhino

An instance of class `Rhino` represents a single rhino. It has several fields that one might use to describe a rhino, as well as methods that operate on these fields. Here are the fields, all of which should be **private** (you can choose the names of these fields).

- name (a `String`) —
- gender (a boolean —`true` for female and `false` for male)
- month of birth (an `int`)

- year of birth (an `int`)
- tag (an `int`)
- father (a `Rhino` object)
- mother (a `Rhino` object)
- number of children (an `int`)
- rhino population (a static `int`)

Here are some details about these fields:

- The name is used to identify the rhino. It can be any string of letters and digits. All rhinos will have different names. Your program should NOT checkthat rhino names are legal.
- The month of birth is in the range 1..12, representing a month from January to December. The year of birth is something like 1857 or 2005. Do not worry about invalid dates; do **not** write code that checks whether dates are valid: assume they are valid.
- The tag is the number on the rhino's tag.This is an integer ≥ 0. If the rhino is not tagged yet, this field contains –1.
- The father and mother fields are the names of the `Rhino` objects that correspond to this rhino's parents. They are `null` if not known.
- The rhino population is the number of rhinos for whom objects (manila folders) have been created. WHENEVER A `RHINO` OBJECT IS CREATED, THIS FIELD SHOULD BE INCREASED BY 1.

Accompanying the declarations of these fields should be comments that describe what each field means —what it contains. For example, on the declaration of field `tag`, one should put that the field is –1 if the rhino is untagged and the tag number itself (≥ 0) if the rhino is tagged. The collection of these fields is called the "class invariant".

Whenever you write a method (see below), look through the class invariant and convince yourself that the class invariant is correct when the method ends, for **all** objects of class `Rhino`. For example, if the method does something to the mother field of the object, are all the mother-object fields correct?

Rhino Methods

Class `Rhino` has the following methods. Pay close attention to the parameters and return values of each method. The descriptions, while informal, are complete.

Constructor	Description
Rhino(String name, boolean female, int month, int year)	Constructor: a new `Rhino`. Parameters are, in order, the name of the rhino, its gender, and the month and year of birth. The new rhino is not tagged, and its parents are not known.
Rhino(String name, boolean gender, Rhino father, Rhino mother, int month, int year)	Constructor: a new `Rhino`. Parameters are, in order, the name of the rhino, its gender, its father and mother, and the month and year of birth. Precondition: father and mother are not null.

Method	Description
getName()	= the name of this rhino (a `String`)
getGender()	= the gender of this rhino (a `boolean`).
getMOB()	= the month in which this rhino was born, in the range 1..12 (an `int`).
getYOB()	= the year in which this rhino was born (an `int`).
getFather()	= (the name of the object representing) the father of this rhino (a `Rhino`).
getMother()	= (the name of the object representing) the mother of this rhino (a `Rhino`).
getNumberChildren()	= the number of children of this rhino (an `int`).
getTag()	= this rhino's tag (–1 if none) (an `int`)
getPopulation()	Static method. = the number of `Rhino` objects created thus far (an `int`).
toString()	= a `String` representation of this `Rhino`. It has to be in a precise format discussed below.

Method	Description
setName(String n)	Set the name of this rhino to `n`.
setGender(boolean g)	Set the gender of the rhino to `g`.
setMOB(int i)	Set the month of birth for this rhino to `i`.
setYOB(int i)	Set the year of birth for this rhino to `i`.
setTag(int t)	Set whether this rhino's tag to t. Precondition: t ≥ 0 and the tag is currently –1.
setFather(Rhino r)	Set this rhino's father to `r` (and increment r's number of children). Precondition: This rhino's father is `null`, `r` is not `null`, and `r` is male.
setMother(Rhino r)	Set this rhino's mother to `r` (and increment r's number of children). Precondition: This rhino's mother is `null`, `r` is not `null`, and `r` is female.
isOlder(Rhino r)	= "this rhino is older than r " (a `boolean`). Precondition: `r` is not null.
areSameAge(Rhino r1, Rhino r2)	Static function. = "`r1` and `r2` are not null and are the same age —i.e. have the same birth date " (a `boolean`).
isBrother(Rhino r)	= "`r` is this rhino's brother" (a `boolean`). Note: rhino A is called the brother of rhino B if the two are different, if A is male, and if they have at least one parent in common. Precondition: `r` is not `null`.
isSister(Rhino r)	= "`r` is this rhino's sister " (a `boolean`). Note: rhino A is called the sister of rhino B if the two are different, if A is female, and if they have at least one parent in common. Precondition: `r` is not `null`.
areSiblings(Rhino r1, Rhino r2)	Static method. = "`r1` and `r2` are not null and `r1` and `r2` are siblings (brothers or sisters)" (a `boolean`).
isMotherOf(Rhino r)	= "this rhino is `r`'s mother" (a `boolean`). Precondition: `r` is not `null`.
isFatherOf(Rhino r)	= "this rhino is `r`'s father" (a `boolean`). Precondition: `r` is not `null`.
isParentOf(Rhino r)	= "this rhino is `r`'s parent" (a `boolean`). Precondition: `r` is not `null`.
areTwins(Rhino r1, Rhino r2)	Static method. = "`r1` and `r2` are not null and `r1` and `r2` are siblings and have the same birth date" (a `boolean`).

Make sure that the names of your methods match those listed above **exactly**, including capitalization. The number of parameters and their order must also match. The best way to ensure this is to copy and paste. Our testing will expect those method name and parameters, so any mismatch will fail during our testing. Parameter names will not be tested —you can change the parameter names if you want.

Each method **must** be preceded by an appropriate specification, as a Javadoc comment. The best way to ensure this is to copy and paste. After you have pasted, be sure to do any necessary editing. For example, the spec does not have to say that a function is static, because that is known from the header of the method. And the spec of a function does not have to say that the function yields a boolean or int or anything else, because that is known from the header of the method.

A precondition should **not** be tested by the method; it is the responsibility of the caller to ensure that the precondition is met. As an example, in method `isMotherOf`, the method body should not test whether `fm` is `null`. However, in function `areSiblings`, the tests for `fm1` and `fm2` not `null` MUST be made.

The number of children of a newly created rhino is 0. Whenever a rhino R is made the mother or father of another rhino, R's number of children should increase by 1.

It is possible for person R1 to be R2's mother, and visa versa, at the same time. We do not check for such strange occurrences.

Function toString
Here is an example of output from function toString:

> "Male rhino Fatso. Tag 34. Born 6/2005. Has 2 children. Father Weighty. Mother unknown."

Here are some points about this output.

1. Exactly one blank separates each piece of information, and the periods are necessary.
2. "Male" or "Female" has to be capitalized.
3. If the mother field or father field is `null`, use "unknown" for its name; otherwise, use the name that appears in the mother or father.
4. In your method body, you may not use an if statement, but you can use a conditional expression —look it up in the index of the CD *ProgramLive.*

Your method bodies should have no if statements. Your method bodies should contain only assignments and return statements. Points will be deducted if **if** statements are used. Further, conditional expressions may be used only in function toString.

Class RhinoTester

How do you know whether class `Rhino` that you are designing is correct? The only way to be sure is to test it, to see if it does what it is supposed to do. It is not enough simply to try out your class `Rhino` in the interactions pane. Every time you write a method for your class `Rhino`, you should also write a couple of tests for it. Further, you should run your collection of tests frequently to make sure that everything works correctly.

Class `RhinoTester` will contain your JUnit test suite; it will perform these testing tasks for you. Make sure that your test suite adheres to the following principles:

- For each method in your class `Rhino`, your test suite should have **at least** one test case that tests that method.
- The more interesting or complex a method is, the more test cases you should have for it. What makes a method 'interesting' or complex can be the number of interesting combinations of inputs that method can have, the number of different results that the method can have when run several times, the different results that can arise when other methods are called before and after this method, and so on.
- Here is one important point. If an argument of a method can be `null`, there should be a test case that has that argument as `null`.
- Test very basic methods early in your test suite; then move on to more complex ones.
- Don't try to test too many things in a single test case. Each test case should test only a couple of conditions.

If a test changes static variables, they will retain their values in later tests. Also, the tests are not necessarily run in the order in which you list them in your test suite. So when testing static variables, record their initial value at the beginning of the test and test that the *change* in the value is what you expect.

How to do this assignment

We suggest that you proceed as follows.

- First, start a new folder on your harddrive that will contain the files for this project. Sart every new project in its own folder.
- Second, write a class `Rhino` using DrJava. In it, declare the fields in class `Rhino`, compiling often as you proceed. Write comments that specify what these fields mean.
- Third,
  - (1) Write the first constructor and all the getter methods of class `Rhino`.
  - (2) Put a method in class `RhinoTester` that tests whether the first constructor and all the getter methods work.
  - (3) Check that the first constructor and all the getter methods work as required. Don't go on to the next step until this is done.
- Fourth, for the second constructor, write it and test it as done for the first constructor.
- Fifth, write function toString and write a method in `RhinoTester` to test it thoroughly.
- Sixth, write each of the setter methods, add a method in `Rhino` to test them, and test them. We suggest writing and testing one method at a time —write a method, put tests for it in class `RhinoTester`, and test it thoroughly; then move on to the next.
- Seventh, add a method to `RhinoTester` to test the comparison methods. Then work on one of the comparison methods at a time: put in its header and specification (as a comment), write the method body, add test cases to the method in RhinoTester, and test and debug until the method works properly.

At each step, make sure all methods are correct before proceeding to the next step. When adding a new method, cut and paste the comment and the header from the assignment handout and then edit the comment.

Other hints and directions

- **Do not** use `if` statements when completing this assignment. For boolean expressions, the operators `&&` (AND), `||` (OR), and ! (NOT) are sufficient to implement all the methods shown above. You will lose points for using `if` statements.
- Some of the `Rhino` methods can be implemented easily by using other `Rhino` methods that you have already created. Look for these cases. Take advantage of them as much as possible.
- Methods `substring`, `toUpperCase`, and `toLowerCase` in class `String` may be useful.
- Remember that a `String` literal is enclosed in double quotation marks and a `char` literal is enclosed in single quotation marks.
- Use method `.equals` to compare objects (including String objects) for equality and `==` to compare primitive values for equality.
- Only object variables can have the value `null`. So comparisons between primitive types and `null` are not legal.
- To create a JUnit test suite, select menu item `File ->?New JUnit Test Case`, and then replace the `testX` method with many methods that test your `Rhino` functionality.

Submitting the assignment

Check these points before you submit your assignment

- Did you use an if statement? Get rid of it.
- Did you make sure that each method is tested enough? For example, if an argument can be null, is there at least one test case that has a call on the method with that argument being null?
- Did you check your javadoc? Click the javadoc button in the DrJava navigation bar. This will cause the specification of the classes and methods of the classes to be extracted from your program and html pages to be created that contain the specs. You should look at those specs carefully and make sure that the specs are suitable. Can you understand precisely what a method does based on the extracted spec? If not, fix the spec, generate the javadoc, and look at it again.

Submit only files that end with the `.java`. Be careful about this, because in the same place as your `.java` files you may also have files that end with `.class` or `.java~`. but otherwise have the same name.