Monitoring Rhinos

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. For more info on endangered species, see http://www.worldwildlife.org/. Some of the rhino pictures in this assignment were downloaded from www.birdingafrica.net.

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. Keep your eyes peeled, and it won't take long for you to see one.

Your task in this assignment is to develop a Java class Rhino, which will maintain information about a rhino, and a JUnit class RhinoTester to maintain a suite of testcases for Rhino. This assignment will help illustrate how Java’s classes and objects can be used to maintain data about a collection of things.

Read the whole assignment before starting. Follow the instructions given below in order to complete the assignment as quickly and as efficiently as possible.

HELP

If you don't know where to start, if you don't understand testing, if you are lost, SEE SOMEONE IMMEDIATELY — the course instructor, a TA, a consultant. Do not wait. 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: 'F' for female and 'M' for male. (a char)
• month of birth. (an int)
• year of birth. (an int)
• tag. (an int)
• father: the name on the folder of the father's Rhino object (null if unknown). (a Rhino)
• mother: the name on the folder of the mother's Rhino object (null if unknown). (a Rhino)
• number of children of this Rhino. (an int)

Here are some details about these fields:

• The name is any string of characters. Like people, rhinos can have the same name.
• 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 \geq 0. If the rhino is not tagged yet, this field contains \(-1\).
• The father and mother fields are the names of the Rhino objects (manila folders) that correspond to this rhino's parents. They are not Strings. They are null if not known.

Class invariant

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, state in a comment that the field is \(-1\) if the rhino is untagged and the tag number itself (\geq 0) if the rhino is tagged. The collection of the comments on these fields, giving the meanings of the fields and constraints on them, 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. And a constructor must assign to fields to make the class invariant true.

**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.

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
</table>
| Rhino(String n, char g, int m, int y) | Constructor: a new Rhino with name n, gender g, birth month m, and birth year y.  
  Precondition: The gender is either 'M' (for male) or 'F' (for female), and the birth month is in 1..12. The new rhino is not tagged, its parents are not known, and it has no children. |
| Rhino(String n, char g,           | Constructor: a new Rhino with name n, gender g, birth month m, birth year y, tag t, and father and mother dad and mom.  
  Precondition: The gender is either 'M' (for male) or 'F' (for female), and |
the birth month is in 1..12. The tag is >= 0, with -1 meaning that the rhino is untagged. dad is not null and its gender is 'M'. mom is not null and its gender is 'F'.

When you write a constructor body, be sure to set all the fields to appropriate values that make the class invariant true.

<table>
<thead>
<tr>
<th>Getter Method</th>
<th>Description</th>
<th>Return type</th>
</tr>
</thead>
<tbody>
<tr>
<td>getName()</td>
<td>= the name of this rhino.</td>
<td>String</td>
</tr>
<tr>
<td>isMale()</td>
<td>= &quot;this rhino is male&quot;.</td>
<td>boolean</td>
</tr>
<tr>
<td>getMOB()</td>
<td>= the month this rhino was born in the range 1..12.</td>
<td>int</td>
</tr>
<tr>
<td>getYOB()</td>
<td>= the year this rhino was born.</td>
<td>int</td>
</tr>
<tr>
<td>getMom()</td>
<td>= the name of (the folder containing) this rhino’s mother.</td>
<td>Rhino</td>
</tr>
<tr>
<td>getDad()</td>
<td>= the name of (the folder containing) this rhino’s father.</td>
<td>Rhino</td>
</tr>
<tr>
<td>getTag()</td>
<td>= this rhino’s tag number (-1 if it has not been assigned one)</td>
<td>int</td>
</tr>
<tr>
<td>getNumberOfChildren()</td>
<td>= the number of rhinos that have this one as a parent.</td>
<td>int</td>
</tr>
<tr>
<td>toString()</td>
<td>= a representation of this Rhino. Its precise specification is discussed below.</td>
<td>String</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Setter Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>setName(String s)</td>
<td>set the name of this rhino to s.</td>
</tr>
<tr>
<td>setGender(char g)</td>
<td>set the gender of this rhino to g ('M' for male, 'F' for female).</td>
</tr>
<tr>
<td>setMOB(int m)</td>
<td>set the month this rhino was born to m</td>
</tr>
<tr>
<td></td>
<td>Precondition: m is in the range 1..12.</td>
</tr>
<tr>
<td>setYOB(int y)</td>
<td>set the year this rhino was born to y.</td>
</tr>
<tr>
<td>setMother(Rhino mom)</td>
<td>set this rhino’s mother to mom.</td>
</tr>
<tr>
<td></td>
<td>Precondition: this rhino’s mother is null and mom is not null.</td>
</tr>
<tr>
<td>setFather(Rhino dad)</td>
<td>set this rhino’s father to dad.</td>
</tr>
<tr>
<td></td>
<td>Precondition: this rhino’s father is null and dad is not null.</td>
</tr>
<tr>
<td>setTag(int t)</td>
<td>set this rhino’s tag number to t.</td>
</tr>
<tr>
<td></td>
<td>Precondition: t is non-negative and has not been assigned to another rhino.</td>
</tr>
</tbody>
</table>
### Comparison Method Description Return type

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Return type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>isOlder(Rhino r)</code></td>
<td>&quot;r is not null and this rhino is older than r&quot;.</td>
<td>boolean</td>
</tr>
<tr>
<td><code>isOlder(Rhino r1, Rhino r2)</code></td>
<td>(Static method) = &quot;r1 and r2 are not null, and r1 is older than r2&quot;.</td>
<td>boolean</td>
</tr>
</tbody>
</table>

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 names, 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, or "spec", 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 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. For example, it is a mistake to call method `setFather` with `null` for the dad argument.

It is possible for `Rhino r1` to be `r2`'s mother, and vice versa, at the same time. We do not check for such strange occurrences.

In writing methods `setFather` (and `setMother`), be careful! If F is becoming the father of this Rhino, then F has one more child, and its field that contains its number of children has to be updated accordingly. The same thing happens with the second constructor.

Do not use if-statements!

You may use conditional expressions (... ? ... : ...) only in function `toString`. Boolean expressions, using the operators `&&` (AND), `||` (OR), and `!` (NOT), are sufficient to implement the comparison methods.

### Function `toString`

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


Here are some points about this output.

1. Exactly one blank separates each piece of information, and the periods are necessary.
2. The 'M' at the beginning means it is a male. Use 'F' for a female.
3. The mother and father are not described in the output of this function.
4. Do not print a negative tag number. If a rhino has a negative tag number, omit all information about its tag entirely. In the above example, " Tag 34. " would be omitted.
5. If a rhino has exactly 1 child, the word "child" should appear instead of "children".
6. In writing the body of function `toString`, do not use if-statements or switches. Instead, use the
conditional expression (condition ? expression-1 : expression-2). This is the only place that you may use the conditional expression.

Class RhinoTester

We require you to build a suite of test cases as you develop class Rhino in a JUnit class RhinoTester. Make sure that your test suite adheres to the following principles:

- Every method in your class Rhino needs at least one test case in the test suite.
- 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.
- Testing should ensure that each part of a method is exercised (executed, evaluated) in at least one test case.
- Here is one important point. If an argument of a method call can be null, there should be a test case that has null for that argument.
- Test each method (or group of methods) as soon as you finish, before moving on to the rest of the assignment.

How to do this assignment

First, remember that if-statements are not allowed. If your assignment has if-statements, you will be asked to revise the assignment and resubmit.

Second, develop and test the class in a methodologically sound way, which we outline below. If we detect that you did not develop it thus, we may ask you to start from scratch on a different assignment.

- First, start a new folder on your hard drive that will contain the files for this project. Start 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. Fill in the class invariant —the comments that give the meanings of the field and constraints on them— as you put in the declarations of the fields.
- Third, after finishing the first section, start a new JUnit class, calling it RhinoTester.
- Fourth, implement and test the methods you write in the groups shown below. For each group, (1) write the methods, (2) write a test procedure in class RhinoTester for the group, (2) add test cases to it for all the methods you wrote, and then (4) test the methods thoroughly. Do not go on to the next group of methods until the ones you are working on are correct. If we determine that you did not follow this way of doing things —for example, you wrote all methods and then tried to test, we may ask you to set this assignment aside and do another instead. Here are the groups of methods.
  - (1) The first constructor and all the getter methods of class Rhino.
  - (2) The setter methods.
  - (3) The second constructor.
  - (4) Function toString.
  - (5) The two comparison methods.

With each group, make sure all methods are correct before proceeding to the next group. When adding a
new method, cut and paste the comment and the header from the assignment handout and then edit the
comment. It must have suitable javadoc specifications as well as suitable comments on the field declarations.
The assignment will not be accepted for testing until it does.

Other hints and directions

- Do not use if statements.
- Remember: a String literal is enclosed in double quotation marks and a char literal is enclosed in single quotation marks.
- Be sure to create the javadoc specification and look at it carefully, to be sure that the methods are specified thoroughly and clearly.

Procedure for submission

You may submit the assignment whenever you wish, as long as it is complete. Do not submit it if you have not completed the test cases or if clicking the Test button results in an error.

1. If the class invariant and the javadoc specifications are not appropriate, we will ask you to fix them.

2. We will look at your test cases. If they are inadequate, we will ask you to fix them.

3. If the test cases are adequate, we will test your program. If there are errors, you will be asked to correct them.

Only when the assignment passes all three categories, will it be accepted as complete.

Submitting the assignment

First, at the top of file Rhino.java, put a comment that says that you looked carefully at the specifications produced by clicking the javadoc button and checked that the specifications of methods and the class specification were OK (put this comment after doing what the comments says).

Second, upload files Rhino.java and RhinoTester.java on the CMS. You will be asked to upload two files: Rhino.java and RhinoTester.java. Make sure you submit .java files. do not submit a file with the extension/suffix .java~. You can ensure that you do this by setting your preferences in your operating system so that extensions always appear.

Have fun!