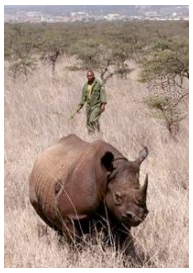
CS1110 Fall 2008 Assignment A1
Submit on the CMS by midnight on Friday, 19 September
Monitoring Rhinos

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. 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. Gries sees them often in his back yard near Community Corners.

This assignment is the first of two that, together, illustrate 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. For this assignment, you will design and implement a class `Rhino`, which is used by to keep track of rhinos, and a class `RhinoTester`, which is a JUnit class to test the methods of class `Rhino`. Pay careful attention to the order in which you write and test class `Rhino`. Follow the instructions given below in order to complete the assignment as quickly and efficiently as possible.

Working with a partner

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. Both people must do something before the group is formed: one proposes, the other agrees. If you do this assignment with another person, you *must* work together. It is against the rules for one person to do some programming on this assignment without the other person sitting nearby and helping. You should take turns "driving" —using the keyboard and mouse.

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. Over 50 of you have never programmed before, and it is reasonable to expect that you may not fully grasp everything immediately. But a little one-on-one help can do wonder.

How this assignment is graded

Because many of you are new to programming, and all of you are new to this course, this assignment will be graded differently than the rest. You can all get 100/100 on it. There will be several steps:

1. Submit your assignment by the deadline —before that if possible. Almost immediately, we will look at the class invariant and javadoc specifications. If they are not appropriate, you will be told what is wrong and asked to fix them and resubmit the assignment within 24 hours.
2. Once the specs are OK, we will look at your test cases to see whether they are adequate. If not, you will be asked to add more test cases and resubmit the assignment within 24 hours.
3. Once the test cases appear to be appropriate, we will execute a program that tests your, looking for errors. If mistakes are found, you will be asked to fix them and resubmit within 48 hours.
4. Step 3 will be repeated until our testing uncovers no more errors.

The goal of all this is (1) to ensure that you learn what is expected on assignments and (2) to enable you to produce a correct program. Should you not get resubmissions done in a reasonable amount of time, some points will be deducted. It would be nice for everyone to be finished with this assignment by a week after it is first due. We will ourselves be reasonable about this.

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 a Rhino object. (of type Rhino)
- mother: the name on the folder of a Rhino object. (of type 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, two or more 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 ≥ 0. If the rhino is not tagged yet, this field contains –1.
- The father and mother fields contain 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.

Accompanying the declarations of these fields should be comments that describe what each field means — what it contains— along with constraints on it. 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 (≥ 0) if the rhino is tagged, and field gender can be only 'F' or 'M'. 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.

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, int month, int year, char gender)	Constructor: a new Rhino. Parameters are, in order, the name of the rhino, the month and year of birth, and its gender. The new rhino is not tagged, its parents are not known, and gender is 'F' or 'M'. It has no children.
Rhino(String name, int month, int year, char gender, Rhino father, Rhino mother, int tag)	Constructor: a new Rhino. Parameters are, in order, the name of the rhino, the month and year of birth, its gender, its father, its mother, and a tag. It has no children. Precondition: father is not null and its gender is 'M'. mother is not null and its gender is 'F'. The tag is >= 0.

When you write a constructor body, be sure to set **all** the fields to appropriate values.

Getter Method	Description
getName()	= the name of this rhino. (a String)
getGender()	= the gender of this rhino ('M' for male, 'F' for female). (a char)
getMOB()	= the month this rhino was born in the range 1..12. (an int)
getYOB()	= the year this rhino was born. (an int)
getMother()	= the name of (the folder containing) this rhinos's mother. (of type Rhino)
getFather()	= the name of (the folder containing) this rhino's father. (of type Rhino)
getTag()	= this rhino's tag number (-1 if it has not been assigned one) (an int)
getNumberOfChildren()	= the number of rhinos that have this one as a parent. (an int)
toString()	= a String representation of this Rhino. Its precise specification is discussed below.

Setter Method	Description
setName(String s)	set the name of this rhino to s.
setGender(char g)	set the gender of this rhino to g ('M' for male, 'F' for female).
setMOB(int m)	set the month this rhino was born to m in the range 1..12
setYOB(int y)	set the year this rhino was born to y.

setMother(Rhino mom)	set this rhino's mother to mom. Precondition: this rhino's mother is null and mom is not null.
setFather(Rhino dad)	set this rhino's father to dad. Precondition: this rhino's father is null and dad is not null.
setTag(int tag)	set this rhino's tag number to tag. Precondition: tag is non-negative and has not been assigned to another rhino.

Comparison Method	Description
isOlder(Rhino r)	= "r is not null and this rhino is older than r". (a boolean)
isOlder(Rhino r1, Rhino r2)	(Static method) = "r1 and r2 are not null, and r1 is older than r2". (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 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. Therefore, delete "(of type Rhino)" in the above spec of getFather.

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, the call "setFather(null);" is a mistake.

It is possible for Rhino r1 to be r2's mother, and visa 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.

Do not use if-statements! Use conditional expression (... ? ... : ...) 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:

"M rhino Fatso. Tag 34. Born 6/2005. Has 2 children."

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). Use a conditional expression *only* in the body of function toString.

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.
- Here is one important point. If an argument of a method 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, you should 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 do it all over again.

- 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. Write comments that specify what these fields mean.
- Third, after finishing the first section, start a new JUnit class, calling it RhinoTester.
- Fourth, this assignment should properly be done in several parts. For each part, write the methods, write a test procedure in class RhinoTester and add test cases to it for all the methods you wrote, and then 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 start the project all over and do it again. Here are the groups of methods.
    - (1) The first constructor and all the getter methods of class Rhino.
    - (2) The second constructor.
    - (3) Function toString.
    - (4) The setter methods.
    - (5) The two comparison methods.

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. 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 that 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

When you are ready to submit, in DrJava, click the javadoc button in order to create html pages that contain the specficatioins of your two classes. Look carefully at all the specs of the methods in class Rhino and make sure they are appropriate. Can you understand precisely WHAT a method does from its specification? If not, the specification is no good and must be changed. After doing this, place a comment at the top of class Rhino, stating that you looked at the javadoc specification and you believe that is is appropriate.

You may submit the assignment whenever you wish. The sooner the better. We will look at it in several steps.

0. If you did not put a comment at the top of class Rhino, indicating that you looked at and corrected the javadoc specifications, or if your javadoc comments are not appropriate, we will ask you to fix this and resubmit.

1. If the field specifications and the javadoc method specifications are not appropriate, we will ask you to fix them within 24 hours and resubmit. Remember, the easiest way to get them right is to copy from this handout and paste (and then edit a bit).

2. If the field and javadoc specs are ok. We will look at your test cases. If they are inadequate, we will ask you to fix them within 24 hours and resubmit.

3. If the test cases are adequate, we will test your program. If there are errors, you will be asked to correct them within 48 hours and resubmit. Step 3 may be repeated several times.

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

Submitting the assignment

Submit classes Rhino and RhinoTester on the CMS. Make sure you submit .java files. Do not submit a file with the extension/suffix .java~ or .class. You can ensure that you do this by setting your preferences in your operating system so that extensions always appear.