Cornell net id _____ **Name** _____

Section day _____ **Section time** _____

CS 1110 Prelim 2 *Grading and posting of grades done Thursday night* 18 March 2010

This 90-minute exam has 6 questions (numbered 0..5) worth a total of 100 points. Spend a few minutes looking at all the questions before beginning. Use the back of the pages if you need more space.

Question 0 (2 points). Fill in the information, legibly, at the top of *each* page. (Hint: do it now.)

Question 1 (12 points). Recall in assignment A1 that procedure `setFather(Rhino dad)` had two preconditions:

(1) this `Rhino` did not yet have a father

(2) `dad` was not null

These two preconditions were there because if-statements could not be used. If the above preconditions were not met, it was likely that a method call would result in execution aborting because of a "null pointer exception" —e.g. something like `null.gender` is evaluated.

Declaration of fields of class Rhino to use in question 1.

`Rhino father`: // father of this rhino, **null** if none

`Rhino mother`: // mother of this rhino, **null** if none

`int` `noc`: // number of children of this `Rhino`

Complete the body of `setFather`, shown below, with those preconditions lifted. Here are things to watch out for. If non-null `Rhino` B is being removed as the father of `Rhino` A, B's number of children should be decreased. If non-null `Rhino` B becomes A's father, B's number of children should be increased. The table to the right gives you the only fields of objects you will have to reference; no methods should be referenced.

```
/** Change this rhino's father to dad.
    Precondition: dad, if not null, is a male.*/
public void setFather(Rhino dad){




}
```

Cornell net id _____ **Name** _____

Section day _____ **Section time** _____

Question 2 Recursion (22 points). It has been determined that our database of rhinos has become corrupted in that some females have been made fathers of rhinos. Write the two procedures specified below, which will help us remove all female fathers. All the methods of `Rhino` and `Vector` that you need are shown in the box to the right.

Methods of class `Rhino`:
`r.getFather()`
`r.getMother()`
`r.getGender()`
`r.setFather(Rhino)`
[answer of question 1]

Methods of class `Vector`:
`v.size()` = # elements of `v`
`v.get(int)` = v[k]

The first method removes all female fathers from the ancestral tree of a `Rhino r`. Think of this as follows. For a non-null `Rhino r`, female fathers have to be removed from `r`'s mother's ancestral tree and `r`'s father's ancestral tree. Then, if `r`'s father is a female, `r`'s father should be set to null, thus removing that father.

The corruption mentioned above may not just be with a single `Rhino` tree. Luckily, we have a database of `Rhinos` —a variable `v` of type `Vector<Rhino>`. All we have to do to remove all the female fathers is to execute the call `remove(v,0);`, provided that the second procedure given below is written. Write the procedure. *Do not use a loop; it must be written recursively*.

These two methods are *NOT* defined in class Rhino.
Remember: v[k..] is the list of elements v[k], v[k+1], …, v[v.size()-1].

```
/** Remove all female fathers from r's ancestral tree.
    Note: r may be null, in which case there is nothing to do. */
public static void remove(Rhino r) {




}
```

```
/** Remove female fathers from all rhinos in ancestral trees of rhinos in v[k..]. */
public static void remove(Vector<Rhino> v, int k) {




}
```

Cornell net id _____ **Name** _____

Section day _____ **Section time** _____

Question 3 (22 points)

(a) In the second procedure of question 2, parameter `v` is of type `Vector<Rhino>`, which means that any element retrieved by function `v.add(…)` is of type `Rhino`. Suppose we make parameter `v` have type `Vector` instead, as in the procedure given below. Write the body of this new method. Write the whole body, but the only part we will grade in this new version is the part that deals with the change due to the different parameter type.

```
/** Each element v[k] may be of any class —e.g. JFrame, Rhino, Color, Point, whatever.
    Remove female fathers from all rhinos in ancestral trees of rhinos in v[k..] —if an object
    v[k] is not a Rhino, don't do anything with it. */
public static void remove(Vector v, int k) {
```

This method id *NOT* defined in class Rhino.

```
}
```

(b) Write the following function equals, *to be placed in class* `Rhino`. Assume that the two fields dealing with a birth date that are declared in class `Rhino` are `yob` (year of birth) and `mob` (month of birth).

```
/** = "r is a Rhino and r's birthdate is the same as this Rhino's birthdate" */
public boolean equals(Object r) {
```

This method IS defined in class Rhino.

```
}
```

Cornell net id _____ **Name** _____

Section day _____ **Section time** _____

Question 4. (22 points). This question uses the two classes defined at the bottom of the page.

(a) In the box to the right, draw variables `a`, `b`, and `d` that are declared below:

Two a; Two b; Two d;

(b) Execute the following assignments, drawing any objects that are created in the space to the right below and storing values as needed in the variables drawn in step (a). When storing a value in a variable, cross off the old value; don't erase it. There is no need to draw frames for method calls.

```
a= new Two(2);
b= new One(3);
d= b;
d.bV= 5;
```

(c) Next, using the results of (b), evaluate the following expressions and write their values to the right of the expressions.

1. `Two.add(a, b)`

2. `Two.add(b, d)`

3. `a instanceof One`

4. `b instanceof One`

5. `b instanceof Two`

6. ((One) `b).altMult()`

```
public class Two {
   public int bV;

   public Two (int n) {
     bV= n;
   }

   public int mult() {
     return 2 * bV;
   }

   public static int add(Two a, Two b) {
     return a.mult() + b.mult();
   }
}
```

```
public class One extends Two {
   public One(int n) {
     super(n + 5);
   }

   public int mult() {
     return bV;
   }

   public int altMult() {
     return super.mult();
   }
}
```

Cornell net id _____ **Name** _____

Section day _____ **Section time** _____

Question 5 (20 points).

(a) Why make a class abstract, and how do you do it?

(b) Why make a method abstract, and how do you do it?

0	_____ out of 02
1	_____ out of 12
2	_____ out of 22
3	_____ out of 22
4	_____ out of 22
5	_____ out of 20
Total	_____ out of 100

(c) To the right are five assignment statements. For each,

(1) State whether it is legal (whether it will compile).

(2) For those that are legal, explain what casting goes on when it is executed, if any, and whether you think that that casting takes any time.

```
double d= 1;
int i= 5.000;
Object ob=  new JFrame();
JFrame jf=  ob;
Animal a= (Animal) ob;
```