**AP Computer Science - Animal Project**

**100 Points**

This assignment requires us to build a complex hierarchy of classes that will demonstrate our understanding of such concepts as encapsulation, inheritance and polymorphism.

**Project Hierarchy/Structure:**

1. You will create a java project called *Animals*.
2. All data fields are to be marked as private. You will need to use getter (accessor) and setter (mutator) methods to access these data fields from outside the class itself.
3. One *Animal* Superclass (One Class to Rule Them All)
   1. You will build one ultimate superclass called *Animal*, which represents the Kingdom Animalia.
   2. Follow my structure for your *Animal* class. (see below)
4. Two Class Subclasses
   1. Underneath Animal, you will build two subclasses, *Bird* and *Mammal*.
   2. These classes (*Bird* and *Mammal*) should inherit all data fields and methods from your *Animal* class.
   3. Follow the below structure for each of these classes.
5. Two Species Class for Class
   1. Underneath *Bird*, you will create a *Penguin* and *Duck* class. These should inherit the *Bird* class info.
   2. Underneath *Mammal*, you will create a *Monkey* and *Tiger*. They inherit Mammal class info.
   3. Again, follow the structure below for the data fields and methods for each of these four classes.
6. You will have a total of 7 classes built. (*Animal, Bird, Mammal, Penguin, Duck, Monkey* and *Tiger*)

**Class Data Fields and Methods**

The following are data fields and methods for each of the seven classes. In the parentheses following each is the primitive data type and its default value.

1. *Animal* (Super-Class)
   1. Data Fields - All data fields should be private, so they cannot be directly accessed outside *Animal*.
      1. *alive* (boolean, true)
      2. *numLegs*(int, 4)
      3. gender (char, 'm')
      4. *reproduction*(String, "eggs"), alternative would be "live birth"
      5. *numAnimals* (static int, 0) - records the total number of animals created
   2. Methods (11 total methods)
      1. *kill*() - changes the *alive* property to *false, reduces numAnimals by 1.*
      2. *setLegs*(int) - Changes the number of legs of the animal to the integer value received.
      3. *setRepro*(String) - Changes the type of reproduction for an animal.
      4. setGender(char)
      5. setNumAnimals(int) - Required for giveBirth Mammal method described below
      6. isAlive() - returns value of alive data field
      7. *get* Methods for all animal data fields (since they must be private) (getNumLegs, getReproduction, getGender, getNumAnimals)
      8. *makeSound*() – Outputs to the console "Makes no sound"
2. *Bird* (subclass of Animal)
   1. Data Fields
      1. *flies* (boolean, true)
      2. *numBirds* - (static, int) - holds the total number of Bird objects created.
   2. Methods (5 total methods)
      1. *canFly*(void) - sets the flies data field to true and outputs "I believe I can fly!".
      2. *cantFly*(void) - sets the flies data field to false and outputs "Grounded again."
      3. getFlies() - returns value of flies data field
      4. *getNumBirds*() - returns number of birds created(*numBirds*)
      5. kill() - reduce numBirds by 1. Should also call Animal kill method.
3. *Mammal* (subclass of *Animal*)
   1. Data Fields
      1. *domestic* (boolean, true) - whether or not it can be kept as a pet.
      2. *value* (int, 500) - cost to acquire this type of mammal.
      3. *numMammals* - (static, int) - holds the total number of *Mammal* objects created.
      4. *hasFingers*(boolean, false) - true/false based on whether they have fingers.
   2. Methods (11 total methods)
      1. *giveBirth*() - 3 possible outcomes here:
         1. If the animal is female, we add 1 to the *numMammals* data field, and output "Another animal is born into the world." (No object is instantiated, just counters incremented)
         2. "Males cannot give birth.", if the gender is male.
         3. "Dead animals cannot give birth." appears if the animal is dead, regardless of gender.
      2. *pickNose*() - Output either "One less booger to deal with." or "You can't pick your nose without fingers." (based on *hasFingers*) or "Dead animals can't pick their nose."
      3. *pet*() - Outputs "You lose your hand trying to pet a non-domestic mammal." or "He/She (depends on gender) enjoys it". (message returned is based on *domestic*) If dead, "No response." is outputted.
      4. kill() - reduces numMammals by 1. Should also call the Animal kill method. (super)
      5. get and set methods are needed for Mammal data fields.
4. *Penguin* - (subclass of Bird)
   1. Data Fields
      1. *appearsInHappyFeet*(boolean, true)
      2. *poleLivesAt*(String, "South")
   2. Methods (6 total methods)
      1. *migrate*() - Moves the penguin from north to south or vice versa (poleLivesAt property changed)
      2. get and set methods are needed for *appearsInHappyFeet* and *poleLivesAt*
      3. sitOnEgg() – Male penguins from the South pole that appear in Happy Feet pole can hatch eggs by sitting on them. Increase appropriate counters and output “Another musically talented penguin comes into the world.”
5. *Duck* - (subclass of Bird)
   1. Data Fields
      1. *hungry(boolean, true)*
      2. *name*(String, "Donald")
   2. Methods (7 total methods)
      1. *findBread*() - If the duck is hungry, set *hungry* to false and output "The duck eats the bread and is no longer hungry." If the duck isn't *hungry*, output "The duck isn't hungry and lets the other ducks feast."
      2. *hunt*() - If name is "Daffy", duck is killed and message "Elmer Fudd got the duck.", otherwise "<*name*> is safe".
      3. get and set Methods are needed for *name* and *hungry*.
      4. makeSound method (see below for more detail)
6. Monkey (subclass of Mammal)
   1. Data Fields
      1. isAlpha - (boolean, false) - can only be set to true if gender is male
      2. iq - (int, 50) - how smart the monkey is.
   2. Methods (8 total methods)
      1. *teach*() - adds 3 to *iq*. Output new *iq* as part of the method. "IQ has risen 3 points and is now at 52.". Dead monkeys can't be taught.
      2. *teach*(int) - adds integer value to *iq*. Output new *iq* as part of the method. "IQ has risen/fallen *x* points and is now at *currentIQ*.". Dead monkeys can't be taught. (overloaded method). Make sure that you account for a negative value being sent in. This could kill the monkey.
      3. *fight*() - If *gender* is male, *isAlpha* is set to *true*, else, "Females cannot be the alpha male." Appropriate message if already Alpha male. "This monkey is already the alpha male."
      4. *hurlPoop*() - hurling poop decreases the *value* of the monkey by $100 each time he hurls poops and decreases *iq* by 1. A monkey cannot have a negative value for *iq* or *value*. If hurling poop causes either *value* or *iq* to go negative, the monkey dies from stupidity or low self-esteem.
         1. If the monkey dies from its value being negative, "This monkey's life is rendered worthless. He dies."
         2. If the monkey dies from 0 iq, "The monkey drops into a coma of stupidity and dies."
         3. Otherwise, "Hurling poop has taken its toll. Value is now $400 and IQ is 49."
      5. get and set Methods are needed for *isAlpha* and *iq*.
      6. makeSound – (see below for more detail)
7. Tiger (subclass of Mammal)
   1. Data Fields
      1. hungry - (boolean, true)
      2. age - (int, 1) - age of tiger
      3. numTigers (static, int) - number of Tigers
   2. Methods (8 total methods )
      1. *hunt() -* Will only hunt if hungry. What is caught is based on age of tiger: (0-3, rabbit), (4-8, gazelle), (9-11, caribou), (12+, blind mountain goat) (Output appropriate prey)
      2. *attackElephant()* - If there are two or more live tigers, the tigers win, otherwise a lone tiger will be killed by elephant. Appropriate messages and changes to data fields are expected.
      3. birthday(int) - tiger ages the number of years sent into the method. If age exceed 15, the tiger dies.
      4. get and set methods are needed for *hungry and age*

**Constructors**

Each class will have at least one constructor. Be certain that in each constructor that you set the inherited data fields to appropriate values. For example, since penguins can't fly, when creating the constructor for *Penguin*, we need to set the inherited *flies* data field to *false*. Be sure to look at all inherited data fields from (Animal, Bird and Mammal). If any of them deviate from the default, change them. Since all data fields are marked as private, you will have to use the get and set methods to access or change any data fields outside of the class.

The complete list of constructors is show below:

**Animal**

Only no-arg (empty) constructor - Set the animal data fields to their default values.

**Bird**

Only no-arg (empty) constructor - Set the *Bird* data fields to their default values. Also, birds have 2 legs and lay eggs.

**Mammal**

Only no-arg (empty) constructor - Set the *Mammal* data fields to their default values. Also, mammals have 4 legs and reproduction should be "live birth".

**Penguin**

Constructor # 1 - No-arg constructor setting data fields to their defaults.

Constructor # 2 - (char, boolean, String) - sets gender (inherited), *appearsInHappyFeet*, and *poleLivesAt*, respectively. Remember that penguins can't fly.

**Duck**

Constructor # 1 - No-arg constructor setting data fields to their defaults.

Constructor # 2 - (char, boolean, String) - sets *gender* (inherited), *hungry* and *name* data fields, respectively.

**Monkey**

Constructor # 1 - No-arg constructor setting data fields to their defaults.

Constructor # 2 - (int, char, boolean, int) - sets the *value* (inherited), *gender* (inherited), *isAlpha*, *iq*

*Monkeys only have 2 legs.* Monkeys also have fingers.

**Tiger**

Constructor # 1 - No-arg constructor setting data fields to their defaults.

Constructor # 2 - (int, char, boolean, int) - sets the *value* (inherited), *gender* (inherited), *hungry*, *age*

Tigers are not domestic.

**makeSound Method**

The makeSound method is inherited from the Animal class and will output to the console the sound that animal makes. Each animal, if it makes sound, should create its own makeSound method to override the Animal one. Here's the list of sounds for each animal:

|  |  |
| --- | --- |
| **Animal** | **Sound** |
| Penguin | Makes no sound |
| Duck | If male, it makes no sound. If female, "Quack". |
| Monkey | If iq is greater than 75, "Hello", otherwise, "Oooh oooh aaah aaah" |
| Tiger | "Roar", unless age is over 13 or under 2, then "Meow". |

\*Dead animals make no sound.

**AnimalRunner**

**Expected Output for Methods**

Animal - makeSound() - "Makes no sound"

Bird - fly() - "I believe I can fly!" or "Dead birds can't fly."

Bird - cantFly() - "Grounded again".

Mammal - giveBirth()

* "Dead animals cannot give birth."
* "Males cannot give birth"
* "Another animal is born into the world."

Mammal - pickNose()

* "Dead animals can't pick their nose."
* "One less booger to deal with."
* "You can't pick your nose without fingers."

Mammal - pet()

* "No response." (if Mammal is dead)
* "You lose your hand trying to pet a non-domesticated mammal."
* "He enjoys it." or "She enjoys it."

Duck - findBread()

* "Dead ducks can't find bread."
* "The duck eats the bread and is no longer hungry."
* "The duck isn't hungry and lets the other ducks feast."

Duck - hunt()

* "Dead ducks aren't hunted."
* "Elmer Fudd got the duck" (if name is "Daffy"
* " <name> is safe"

Duck - makeSound()

* "Quack" (if female and alive), otherwise "Makes no sound".

Monkey - teach()

* "Can't teach dead monkeys"
* "IQ has risen 3 points and is now at <iq>"

Monkey - fight()

* "Dead monkeys cannot become the alpha male."
* "We have a new alpha male."
* "Females cannot be the alpha male."

Monkey - hurlPoop()

* "Dead monkeys cannot hurl poop."
* "Hurling poop has taken its toll. Value is now <value -100> and iq drops to <iq -1>
* "The monkey's life is rendered worthless. It dies." (when value gets to 0)
* The monkey drops into a coma of stupidity and dies. (when iq gets to 0)

Monkey - makeSound()

* "Hello" (if iq is greater than 75)
* "Oooh oooh aaah aaah"

Tiger - hunt()

* "Dead tigers don't hunt."
* "The tiger isn't in a hunting mood due to lack of hunger."
* "The tiger captures and eats a rabbit" (age 3 or less)
* "The tiger captures and eats a gazelle" (age 4-8)
* "The tiger captures and eats a caribou" (age 9-11)
* "The tiger captures and eats a blind mountain goat" (age 12+)

Tiger - attackElephant()

* "Dead tigers don't attack elephants."
* "<numTigers> gang up and overcome the elephant and feast like tigers." (if numTigers > 1)
* "The foolish tiger tries to take down the elephant on his own and perishes from a tusk-induce lung puncture."

Tiger - birthday(int a)

* "Dead tigers don't celebrate birthdays."
* "The tiger ages <a> years. It is now <age> years old."
* "The tiger dies of old age." (If older than 15).

Tiger - makeSound()

* "Roar" (if age is 2 or more)
* "Meow" (if age is less than 2)

**Animal Kingdom Laws**

* Killing any animal should decrease the static *numAnimals* data field and appropriate other static data fields. (*numMammals*, *numBirds*, *numTigers*, etc…)
* Dead animals cannot do anything. Appropriate messages should appear when a dead animal is asked to perform an action.