AnimalSim (Java)

General Description

A basic application with the purpose of highlighting concepts of object-oriented programming and inheritance.

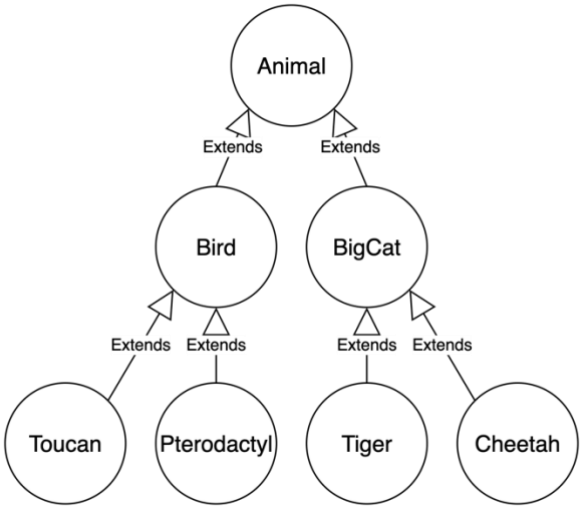
AnimalSim allows you to pick and create animal objects from a selection of seven different animal object types:

- Animal
- Bird
- BigCat
- Toucan
- Pterodactyl
- Tiger
- Cheetah

The animal objects have three levels of hierarchy.

- *Animal* is the parent class for all animal objects.
- The classes *Bird* and *BigCat* are child classes of *Animal*.
- The classes *Toucan* and *Pterodactyl* are child classes of *Bird*.
- The classes *Tiger* and *Cheetah* are child classes of *BigCat*.

The child classes inherit most of their attributes from their parent classes, with the addition of 1-2 unique additional attributes.



Things that can be done

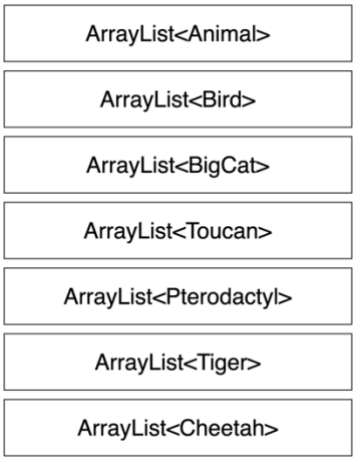
In addition to creating the animal objects, a small variety of other functions can be performed through the application to handle the data generated.

The application's menu gives the following options:

- Creating an animal
- Deleting an animal
- Displaying all of your created animals
- Searching for an animal
- Printing all created animals to a text file
- Creating animals by reading from a text file
- Exit

Driver Program

An ArrayList is created for each animal object type and used to store and access all objects of that type for all menu functions.



The menu is displayed and will be redirected to after every time a menu option is performed, until the program is exited.

Menu options:

- **Creating an animal**
  - A list is shown to choose which type of animal object you would like to create.
  - After choosing, the user input is asked for each property for that animal object type.
  - The animal is created and added to its list.

- **Deleting an animal**
  - Deleting can be done in two different ways; by index or by searching.

- Index-deleting will simply ask which type of animal you would like to delete, and then the index number to be deleted in that animal's array. If no animal exists for the index chosen, an error message will be given.

- Search-deleting will ask which type of animal you would like to delete, and then ask for all of the information of the specific animal you would like to delete, in order to find and remove it. With search-deleting, all of the information needs to be entered exactly, case-sensitive included, for the animal to be found and removed.

  If no animal is found that matches the information, an error message will be given and the search criteria displayed.

  If the animal is found, it will be removed, and the index it was removed from will be displayed.

- **Displaying all of your created animals**
  - All animal object types are listed and the existing animals in the type's list are printed to console.

- **Searching for an animal**
  - Searching for an animal is the same as the searching process in search-deleting. The user is asked which animal type they would like to search for, following entering all of the property fields for that animal. A separate instance of the animal is created and used to compare and search for the actual animal. The animal's property values must be entered exactly, case-sensitive included, for the animal to be found.
  - The search will return all elements in the list that match the search criteria.
  - If no match is found, a message is displayed saying there was not match and the search values are displayed.

- **Printing all created animals to a text file**
  - The user can choose which directory to save the text file in - The first time the user chooses this option, they will be asked for the pathname of the directory to save the text file to. If the user chooses this option again later in the same session, they will not be asked for the directory again and the text file will be saved to the same directory.

  - Text file naming will handle duplicate file names if they already exist in the directory -

    Text files are saved with the naming convention,

    AnimalObjects#

    where the # is a number that will be incremented if a file with the same name already exist. The # will start at one (1) and increment by one if the filename already exists and checked again. It will continue incrementing and checking until a filename is reached that does not already exist in the directory.

    Printing to a text file multiple times will create new text files with the incremented names.

- An error message will be displayed each time a text file fails to be created if the filename already exists in the directory. It will say that there may already be another file with the same name and display the entire directory pathname attempted (including the file name).
- When a text file is successfully created, a message will be displayed saying that the process was successful and the full directory pathname for the file will be displayed.

- **Creating animals by reading from a text file**

  - **Constraint:** In order to be successfully read and created, the file must have the name type of the object to be created on a line by itself: "Animal", "Bird", "BigCat", "Toucan", "Pterodactyl", "Tiger", or "Cheetah". Any misspelling of the name of additional characters to the name on the same line won't be recognized.

  - **Constraint:** Furthermore, all properties will need to be listed, each on a separate line in the correct specific order. The order is the same order that the constructors follow, and the user is asked when creating that type of object. Any deviation from that order will result in the animal not being recognized.

  - The correct way to enter information in the text file for reading goes as follows: Only the first column is necessary, the second column is only to show the data type

Object-type	
name	String
location	String
size	float
age	int
unique property (if applicable)	depends
unique property (if applicable)	depends

  - After successfully creating animals from a text file, the total amount of animals created for each animal object type will be displayed.

- **Exit**
  - The program is exited, and all the session's data is deleted.