

Putting it all in **Zoo**sЯus

1 Purpose

The purpose of the final project is to summarize A3a&b and A4a&b into an application that showcases OOP concepts of abstraction, inheritance, polymorphism as well as Java concepts of interfaces and abstract classes, text file processing and exception handling.

2 Your Task

Pete Pettington lives in a farm taking care of his pets, which consists of a number of dogs, cats, and ducks. He owns a startup named **Zoo**sЯus that specializes in renting out petting zoos to the local shopping malls.

In this assignment, you will write a program to help Pete with his tedious and time consuming task of keeping track of his pets and the shopping malls that host them.



3 Input

Your program will allow the user to build a database of text files, each storing records for the pets in a particular petting zoo, and each named after the shopping mall hosting the petting zoo.

The database files are comma-separated values (CSV) files, using exact same format your are already familiar with from assignments 3 and 4.

For example, contents of a database file named **Quartier_Cavendish.txt**¹ may look like this:

```
cat, yes , Alex , 5 , M
duck, 1, Munchkin, 5 ,f
Cat,no,Sam , 7 , m
dog ,Yorkshire Terrier , Luna , 11 , f
Cat , yes , PreciousMissy,11,F
Duck , 4 ,Eggbert,4 , F
Dog, Bernese Mountain ,Jack , 7, m
```

Each line in the file represents a pet record, where the first value indicates the pet type and the second value is specific to that type, such as **breed** for dog, **yes** or **no** for cat being neutered, or an integer for duck eggs. The last three values list the name, age, and gender of the pet, in that order.

¹The file extension may be anything other than **txt** such as **pet**, **zoo**, etc. The only reason we use **txt** for file extension is that most of us use Windows OS, where the default file name extension for a text file is **txt**.

4 Output

Your program should present the user with a neatly formatted output as shown below:

----- Quartier Cavendish Petting Zoo -----					
ID	Pet Type	Name	Age	Gender	Specifics
1	Cat	Alex	5	M	neutered
2	Duck	Munchkin	5	F	1 egg
3	Cat	Sam	7	M	not neutered
4	Dog	Luna	11	F	Yorkshire Terrier
5	Cat	PreciousMissy	11	F	neutered
6	Duck	Eggbert	4	F	4 eggs
7	Dog	Jack	7	M	Bernese Mountain

dogs: 2, cats: 3, ducks: 2					

The **ID** numbers are unique integers that are generated and assigned as the input pet records are each processed (see [Step 4.](#))

5 User Interface

To manage the database, your program must present the user with the following menu of operations, read the user's choice of operation, and perform that operation.

```
Choose from these options
-----
1 - Print pet list
2 - Add a new pet record
3 - Remove a pet record
4 - Sort pet list by name
5 - Sort pet list by age
6 - Sort pet list by gender
7 - Load database from file
8 - Save database to file
9 - Quit

Your choice?
```

Your program will repeat this text-based menu-driven interactive process until the user chooses to quit.

6 Sample Program Run

User input is highlighted in yellow.

```
1 Choose from these options
2 -----
3 1 - Print pet list
4 2 - Add a new pet record
5 3 - Remove a pet record
6 4 - Sort pet list by name
7 5 - Sort pet list by age
8 6 - Sort pet list by gender
9 7 - Load database from file
10 8 - Save database to file
11 9 - Quit
12
13 Your choice? 2
14 Enter 5 comma-separated values: pet type, specific pet info, name, age, gender
15 Dog,Jack Russell Terrier ,Coco, 7 ,f
16 Choose from these options
17 -----
18 1 - Print pet list
19 2 - Add a new pet record
20 3 - Remove a pet record
21 4 - Sort pet list by name
22 5 - Sort pet list by age
23 6 - Sort pet list by gender
24 7 - Load database from file
25 8 - Save database to file
26 9 - Quit
27
28 Your choice? 2
29 Enter 5 comma-separated values: pet type, specific pet info, name, age, gender
30 Duck , 6 ,Nibbles , 3, F
31 Choose from these options
32 -----
33 1 - Print pet list
34 2 - Add a new pet record
35 3 - Remove a pet record
36 4 - Sort pet list by name
37 5 - Sort pet list by age
38 6 - Sort pet list by gender
39 7 - Load database from file
40 8 - Save database to file
41 9 - Quit
```

```

42
43 Your choice? 1
44 -----
45               Quartier Cavendish Petting Zoo
46 -----
47 ID  Pet Type      Name      Age Gender  Specifics
48 -----
49 1    Dog          Coco       7    F    Jack Russell Terrier
50 2    Duck         Nibbles    3    F           6 eggs
51 -----
52 dogs: 1,  cats: 0,  ducks: 1
53 -----
54
55 Choose from these options
56 -----
57 1 - Print pet list
58 2 - Add a new pet record
59 3 - Remove a pet record
60 4 - Sort pet list by name
61 5 - Sort pet list by age
62 6 - Sort pet list by gender
63 7 - Load database from file
64 8 - Save database to file
65 9 - Quit

```

```

67 Your choice? 3
68 Enter the ID number of the pet to remove: 1
69
70 Choose from these options
71 -----
72 1 - Print pet list
73 2 - Add a new pet record
74 3 - Remove a pet record
75 4 - Sort pet list by name
76 5 - Sort pet list by age
77 6 - Sort pet list by gender
78 7 - Load database from file
79 8 - Save database to file
80 9 - Quit
81
82 Your choice? 1

```

```

83 -----
84               Quartier Cavendish Petting Zoo
85 -----
86 ID  Pet Type      Name      Age Gender  Specifics
87 -----
88  2   Duck         Nibbles    3    F      6 eggs
89 -----
90 dogs: 0,  cats: 0,  ducks: 1
91 -----
92
93 Choose from these options
94 -----
95  1 - Print pet list
96  2 - Add a new pet record
97  3 - Remove a pet record
98  4 - Sort pet list by name
99  5 - Sort pet list by age
100  6 - Sort pet list by gender
101  7 - Load database from file
102  8 - Save database to file
103  9 - Quit

```

```

105 Your choice? 3
106 Enter the ID number of the pet to remove: 1
107 Error: no pet with ID number 1
108
109 Choose from these options
110 -----
111  1 - Print pet list
112  2 - Add a new pet record
113  3 - Remove a pet record
114  4 - Sort pet list by name
115  5 - Sort pet list by age
116  6 - Sort pet list by gender
117  7 - Load database from file
118  8 - Save database to file
119  9 - Quit
120
121 Your choice? 2

```

```

122 Enter 5 comma-separated values: pet type, specific pet info, name, age, gender
123 Duck , 3,Wiggles , 10,F
124 Choose from these options
125 -----
126 1 - Print pet list
127 2 - Add a new pet record
128 3 - Remove a pet record
129 4 - Sort pet list by name
130 5 - Sort pet list by age
131 6 - Sort pet list by gender
132 7 - Load database from file
133 8 - Save database to file
134 9 - Quit
135
136 Your choice? 1
137 -----
138               Quartier Cavendish Petting Zoo
139 -----
140 ID  Pet Type      Name      Age Gender      Specifics
141 -----
142 2    Duck        Nibbles      3    F        6 eggs
143 3    Duck        Wiggles     10    F        3 eggs
144 -----
145 dogs: 0,  cats: 0,  ducks: 2
146 -----
147
148 Choose from these options
149 -----
150 1 - Print pet list
151 2 - Add a new pet record
152 3 - Remove a pet record
153 4 - Sort pet list by name
154 5 - Sort pet list by age
155 6 - Sort pet list by gender
156 7 - Load database from file
157 8 - Save database to file
158 9 - Quit

```

```

159
160 Your choice? 3
161 Enter the ID number of the pet to remove: 2
162
163 Choose from these options
164 -----
165 1 - Print pet list
166 2 - Add a new pet record
167 3 - Remove a pet record
168 4 - Sort pet list by name
169 5 - Sort pet list by age
170 6 - Sort pet list by gender
171 7 - Load database from file
172 8 - Save database to file
173 9 - Quit
174
175 Your choice? 1
176 -----
177               Quartier Cavendish Petting Zoo
178 -----
179 ID  Pet Type      Name      Age Gender  Specifics
180 -----
181 3    Duck        Wiggles    10    F        3 eggs
182 -----
183 dogs: 0,  cats: 0,  ducks: 1
184 -----
185
186 Choose from these options
187 -----
188 1 - Print pet list
189 2 - Add a new pet record
190 3 - Remove a pet record
191 4 - Sort pet list by name
192 5 - Sort pet list by age
193 6 - Sort pet list by gender
194 7 - Load database from file
195 8 - Save database to file
196 9 - Quit

```

```

197
198 Your choice? 7
199 Enter input file name: Quartier_Cavendish.txt
200 Completed processing 7 pet records
201 Choose from these options
202 -----
203 1 - Print pet list
204 2 - Add a new pet record
205 3 - Remove a pet record
206 4 - Sort pet list by name
207 5 - Sort pet list by age
208 6 - Sort pet list by gender
209 7 - Load database from file
210 8 - Save database to file
211 9 - Quit
212
213 Your choice? 1
214 -----
215               Quartier Cavendish Petting Zoo
216 -----
217 ID  Pet Type      Name      Age Gender  Specifics
218 -----
219 3   Duck          Wiggles    10    F        3 eggs
220 4   Cat           Alex       5     M        neutered
221 5   Duck          Munchkin   5     F        1 egg
222 6   Cat           Sam        7     M        not neutered
223 7   Dog           Luna       11    F        Yorkshire Terrier
224 8   Cat           PreciousMissy 11    F        neutered
225 9   Duck          Eggbert    4     F        4 eggs
226 10  Dog           Jack       7     M        Bernese Mountain
227 -----
228 dogs: 2,  cats: 3,  ducks: 3
229 -----
230
231 Choose from these options
232 -----
233 1 - Print pet list
234 2 - Add a new pet record
235 3 - Remove a pet record
236 4 - Sort pet list by name
237 5 - Sort pet list by age
238 6 - Sort pet list by gender
239 7 - Load database from file
240 8 - Save database to file
241 9 - Quit

```



```

242
243 Your choice? 4
244 Database is now sorted by name
245 Choose from these options
246 -----
247 1 - Print pet list
248 2 - Add a new pet record
249 3 - Remove a pet record
250 4 - Sort pet list by name
251 5 - Sort pet list by age
252 6 - Sort pet list by gender
253 7 - Load database from file
254 8 - Save database to file
255 9 - Quit
256
257 Your choice? 1
258 -----
259               Quartier Cavendish Petting Zoo
260 -----
261 ID  Pet Type      Name      Age Gender  Specifics
262 -----
263 4    Cat          Alex       5    M      neutered
264 9    Duck         Eggbert    4    F      4 eggs
265 10   Dog          Jack       7    M      Bernese Mountain
266 7    Dog          Luna      11   F      Yorkshire Terrier
267 5    Duck         Munchkin  5    F      1 egg
268 8    Cat          PreciousMissy 11   F      neutered
269 6    Cat          Sam       7    M      not neutered
270 3    Duck         Wiggles   10   F      3 eggs
271 -----
272 dogs: 2,  cats: 3,  ducks: 3
273 -----
274
275 Choose from these options
276 -----
277 1 - Print pet list
278 2 - Add a new pet record
279 3 - Remove a pet record
280 4 - Sort pet list by name
281 5 - Sort pet list by age
282 6 - Sort pet list by gender
283 7 - Load database from file
284 8 - Save database to file
285 9 - Quit

```

```

286
287 Your choice? 5
288 Database is now sorted by age
289 Choose from these options
290 -----
291 1 - Print pet list
292 2 - Add a new pet record
293 3 - Remove a pet record
294 4 - Sort pet list by name
295 5 - Sort pet list by age
296 6 - Sort pet list by gender
297 7 - Load database from file
298 8 - Save database to file
299 9 - Quit
300
301 Your choice? 1
302 -----
303               Quartier Cavendish Petting Zoo
304 -----
305 ID  Pet Type      Name      Age Gender      Specifics
306 -----
307 9   Duck          Eggbert    4    F          4 eggs
308 4   Cat           Alex       5    M          neutered
309 5   Duck          Munchkin   5    F          1 egg
310 10  Dog           Jack       7    M          Bernese Mountain
311 6   Cat           Sam        7    M          not neutered
312 3   Duck          Wiggles    10   F          3 eggs
313 7   Dog           Luna       11   F          Yorkshire Terrier
314 8   Cat           PreciousMissy 11   F          neutered
315 -----
316 dogs: 2,  cats: 3,  ducks: 3
317 -----
318
319 Choose from these options
320 -----
321 1 - Print pet list
322 2 - Add a new pet record
323 3 - Remove a pet record
324 4 - Sort pet list by name
325 5 - Sort pet list by age
326 6 - Sort pet list by gender
327 7 - Load database from file
328 8 - Save database to file
329 9 - Quit

```

```

330
331 Your choice? 6
332 Database is now sorted by gender
333 Choose from these options
334 -----
335 1 - Print pet list
336 2 - Add a new pet record
337 3 - Remove a pet record
338 4 - Sort pet list by name
339 5 - Sort pet list by age
340 6 - Sort pet list by gender
341 7 - Load database from file
342 8 - Save database to file
343 9 - Quit
344
345 Your choice? 1
346 -----
347               Quartier Cavendish Petting Zoo
348 -----
349 ID  Pet Type      Name      Age Gender      Specifics
350 -----
351 9    Duck      Eggbert      4    F      4 eggs
352 5    Duck      Munchkin      5    F      1 egg
353 3    Duck      Wiggles      10   F      3 eggs
354 7    Dog       Luna      11   F      Yorkshire Terrier
355 8    Cat      PreciousMissy  11   F      neutered
356 4    Cat      Alex      5    M      neutered
357 10   Dog      Jack      7    M      Bernese Mountain
358 6    Cat      Sam      7    M      not neutered
359 -----
360 dogs: 2,  cats: 3,  ducks: 3
361 -----
362
363 Choose from these options
364 -----
365 1 - Print pet list
366 2 - Add a new pet record
367 3 - Remove a pet record
368 4 - Sort pet list by name
369 5 - Sort pet list by age
370 6 - Sort pet list by gender
371 7 - Load database from file
372 8 - Save database to file
373 9 - Quit

```

```
374
375 Your choice? 8
376 Enter output file name: Quartier_Cavendish_out.txt
377 Choose from these options
378 -----
379 1 - Print pet list
380 2 - Add a new pet record
381 3 - Remove a pet record
382 4 - Sort pet list by name
383 5 - Sort pet list by age
384 6 - Sort pet list by gender
385 7 - Load database from file
386 8 - Save database to file
387 9 - Quit
```

```
388
389 Your choice? 9
390 goodbye!
```

Saved database Quartier_Cavendish_out.txt

```
1 Duck, 4, Eggbert, 4, F
2 Duck, 1, Munchkin, 5, F
3 Duck, 3, Wiggles, 10, F
4 Dog, Yorkshire Terrier, Luna, 11, F
5 Cat, yes, PreciousMissy, 11, F
6 Cat, yes, Alex, 5, M
7 Dog, Bernese Mountain, Jack, 7, M
8 Cat, no, Sam, 7, M
```

Notice that the id numbers are not written to the output file.

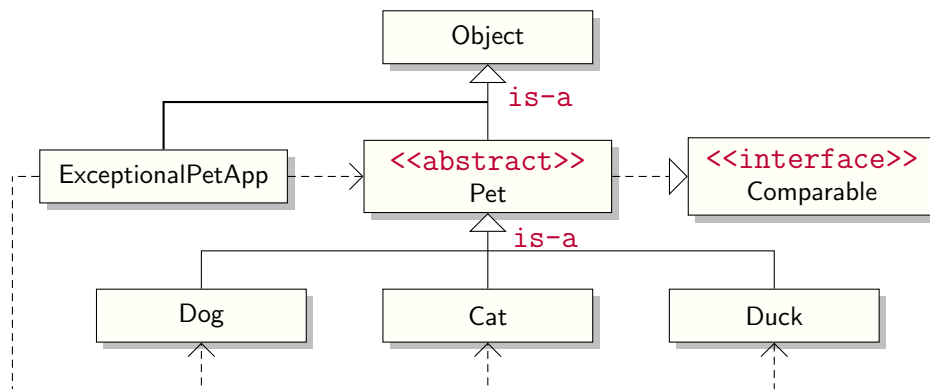
7 Database Storage

Your program must use an `ArrayList<Pet>` to store the database, which is empty at the start of the program.

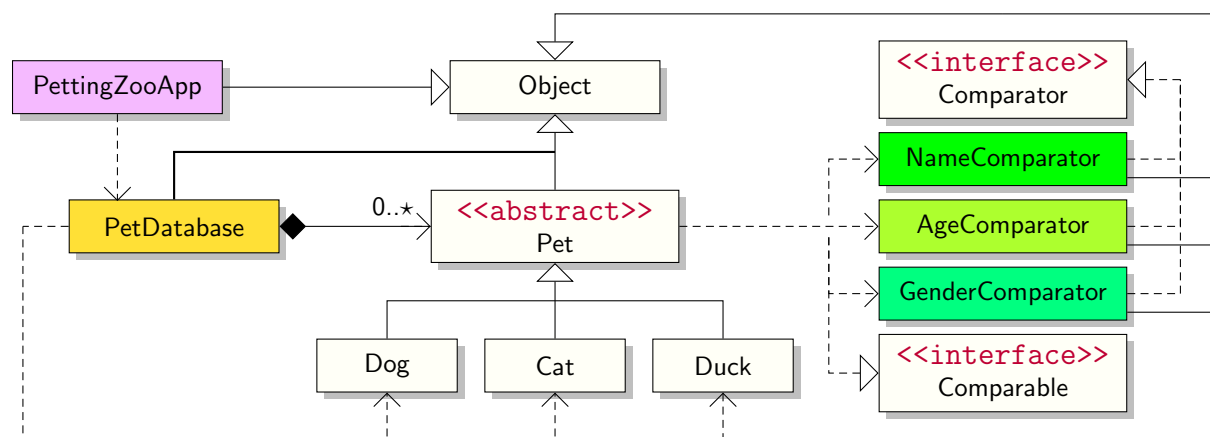
8 Hints

Step 1. Prepare Your Existing Pet Inheritance Hierarchy from 4B

Here is the collapsed version of the UML diagram given on page 6 of assignment 4B, with yellow and green highlights removed:

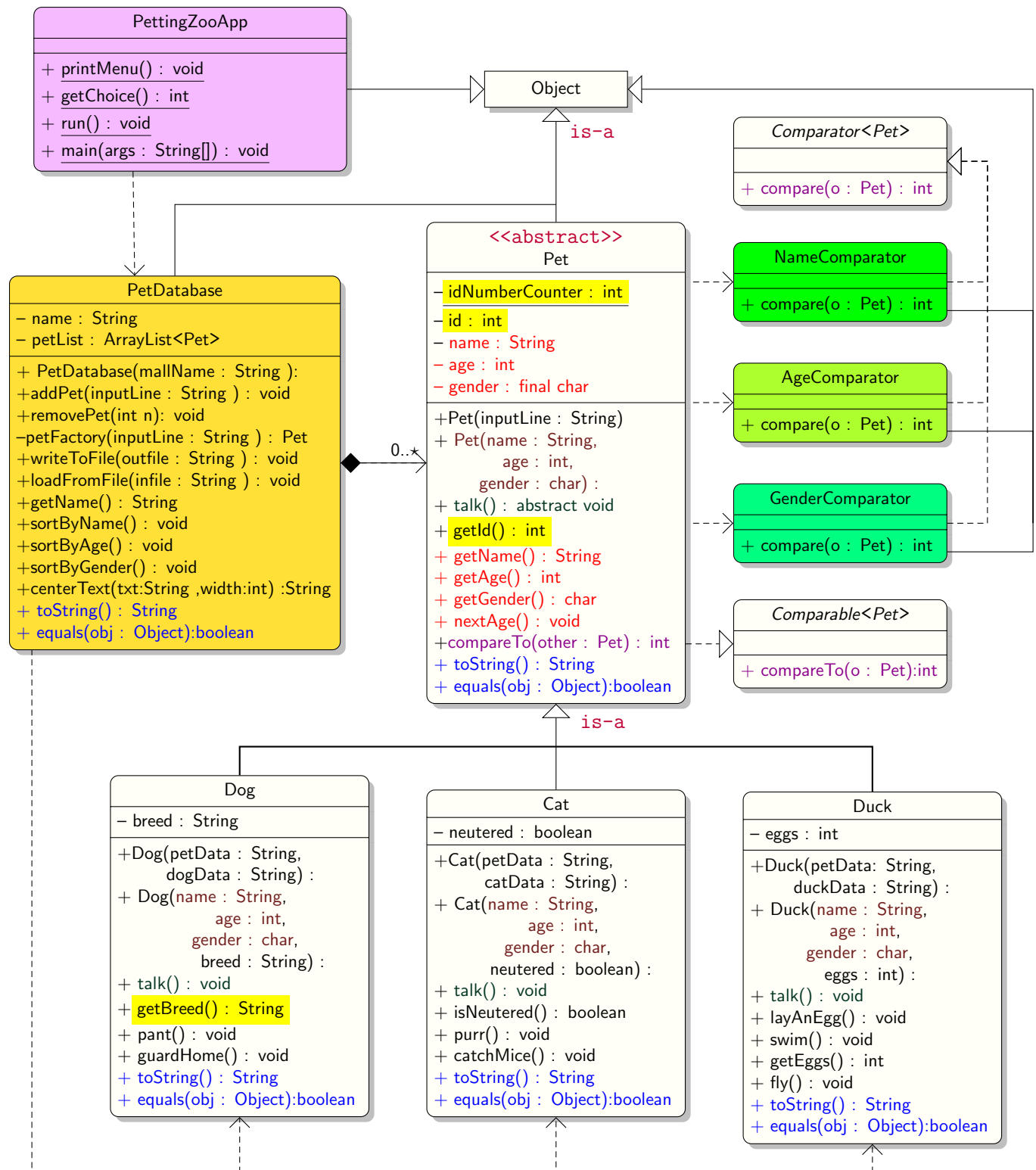


Step 2. Modify the UML diagram from step 1 Removing `ExceptionalPetApp` and replacing it with two new classes `PetDatabase` and `PettingZooApp`, you will implement the class diagram below, which is followed by a quick summary of its meaning:



- Zero or more `Pet` object references are owned by a `PetDatabase` object
- `PettingZooApp` uses `PetDatabase` and extends `Object`
- `NameComparator`, `AgeComparator`, and `GenderComparator` each implement the `Comparator` interface
- `Pet` uses `NameComparator`, `AgeComparator`, and `GenderComparator`
- `Pet` extends `Object` and implements the `Comparable` interface
- Classes `Dog`, `Cat`, and `Duck` extend `Pet` into concrete classes, knowing absolutely nothing about their client `PetDatabase`
- A `PetDatabase` owns zero or more `Pet` object references and depends on `Dog`, `Cat`, and `Duck` objects

Step 3. Look at the big picture and plan the steps to get there



Step 4. Create unique ID for Pet objects Add the following lines 3-5 to your **Pet** class: (back to page 2)

```

1 public abstract class Pet implements Comparable<Pet>
2 {
3     private static int idNumberCounter = 0; // all Pet objects can use this;
4     private int id = ++idNumberCounter;    // assign unique id to this object
5     public int getId() { return id; }
6     // remaining code not shown for brevity
7 }

```

Step 5. Class PetDatabase

PetDatabase	
– name : String	The name of this class stores the name of this pet database
– petList : ArrayList<Pet>	stores records of all pets in this database
+ PetDatabase(name : String):	Initializes this name and creates an empty petList
– petFactory(inputLine : String) : Pet	same as lines 117-159, page 14, assignment 4B (without the static keyword)
+ addPet(inputLine : String) : void	same as lines 38-78, page 12, assignment 4B
+ removePet(int id): void	searches petList for a pet record with the supplied id; if found, removes it; otherwise displays a message, changing nothing; see lines 67 (page 5), 105 (page 6), and 160 (page 8) on this assignment
+ writeToFile(outfile : String) : void	see page 17 on this assignment
+ loadFromFile(infile : String) : void	see lines 17-79, pages 11-12, assignment 4B
+ getName() : String	return this name
+ sortByName() : void	see assignment 3B, Step 1 on page 9, and Step 5 on page 10 (line 13)
+ sortByAge() : void	see assignment 3B, Step 2 on page 9, and Step 5 on page 10 (line 19)
+ sortByGender() : void	similar to sortByName() and sortByAge() above
+ <u>centerText(txt:String,width:int):String</u>	We wrote this method in class; see a version on page 18
+ toString() : String	returns a neatly formatted string representing this object; to get the names of the pet types use the instanceof operator similar to what opertor method on page 17 does the pet types
+ equals(obj : Object):boolean	returns true if this and Pet version of obj have exact same name and exact same petList records

Step 6. Class PettingZooApp

PettingZooApp	The name of this class
+ printMenu() : void	displays a menu of options; see page 2
+ getChoice() : int	reads and returns user's choice from the keyboard taking care of all potential input error
+ main(args : String[]) : void	calls run() below
+ run() : void	repeatedly displays the menu, reads user's choice of operation, and performs that operation

To see what method `run()` does for a given choice see

Choice	1	2	3	4	5	6	7	8	9
Page	4	4	5	9	10	11	8	12	13
Line #	43	28	67 or 105	243	287	331	198	375	389

Step 7. Done!

8.1 void writeToFile(String inFileName)

This is essentially the same as the `writeArrayListToTextFile` method in assignment 3A, page 4, lines 21-31. The only difference here is that we need to fill the first value in each output line with a pet type, namely, "dog", "cat", or "duck". See how the `instanceof` operator is used to make that determination.

```
1 public void writeToFile(String outfileName) throws FileNotFoundException
2 {
3     File file = new File(inFileName);
4     PrintWriter printWriter = new PrintWriter(file);
5     String petType = "unknown type";
6     String petSpecificValue = "unknown specifics";
7     for (Pet pet : petList)
8     {
9         if (pet instanceof Dog)
10        {
11            Dog dog = (Dog) pet;
12            petSpecificValue = "Dog, " + dog.getBreed();
13        }
14        else if (pet instanceof Cat)
15        {
16            Cat cat = (Cat) pet;
17            petSpecificValue = "Cat, " + (cat.isNeutered() ? "yes" : "no");
18        }
19        else if (pet instanceof Duck)
20        {
21            Duck duck = (Duck) pet;
22            petSpecificValue = "Duck, " + duck.getEggs() ;
23        }
24        printWriter.println(petSpecificValue + ", " + pet.getName() + ", "
25                           + pet.getAge() + ", " + pet.getGender());
26    }
27    printWriter.close();
28 }
```

Food for thought: what if there were 30 pet types instead of 3?

8.2 static String centerText(String text, int width)

```
1 //  centers a given text within a field of width 'width'
2 //  centers a given text within a field of width 'width'
3 private static String centerText(String text, int width)
4 {
5     if(width < text.length()) return text; // nothing to do; text is longer than width
6
7     final String BALNK = " ";
8     int total_lead_and_trail_spaces = width - text.length();
9     int lead_spaces =(total_lead_and_trail_spaces)/2;
10    // just in case the total number of leading and trailing spaces is odd
11    int trail_spaces = (total_lead_and_trail_spaces % 2 == 1) ? lead_spaces + 1 :
12                                                              lead_spaces;
13
14    String result = "";
15    // start with the leading spaces
16    for (int i = 0; i < lead_spaces; i++) result += BALNK;
17    result += text; // now the text
18    // end with the trailing spaces
19    for (int i = 0; i < trail_spaces; i++) result += BALNK;
20    return result;
21 }
```

9 Evaluation Criteria

Evaluation Criteria		
Functionality	Ability to perform as required, producing correct output for any set of input data, Proper implementation of all specified requirements, Efficiency	60%
Robustness	Ability to handle input data of wrong type or invalid value	10%
OOP style	Encapsulating only the necessary data inside objects, Information hiding, Proper use of Java constructs and facilities.	10%
Documentation	Description of purpose of program, Javadoc comment style for all methods and fields, comments on non-trivial steps in all methods	10%
Presentation	Format, clarity, completeness of output, user friendly interface	5%
Code readability	Meaningful identifiers, indentation, spacing, localizing variables	5%

This project is the last assignment of the semester.

Good luck in all your endeavors.