



CS 210 Wildlife Zoo Interface Guide

Functional Requirements

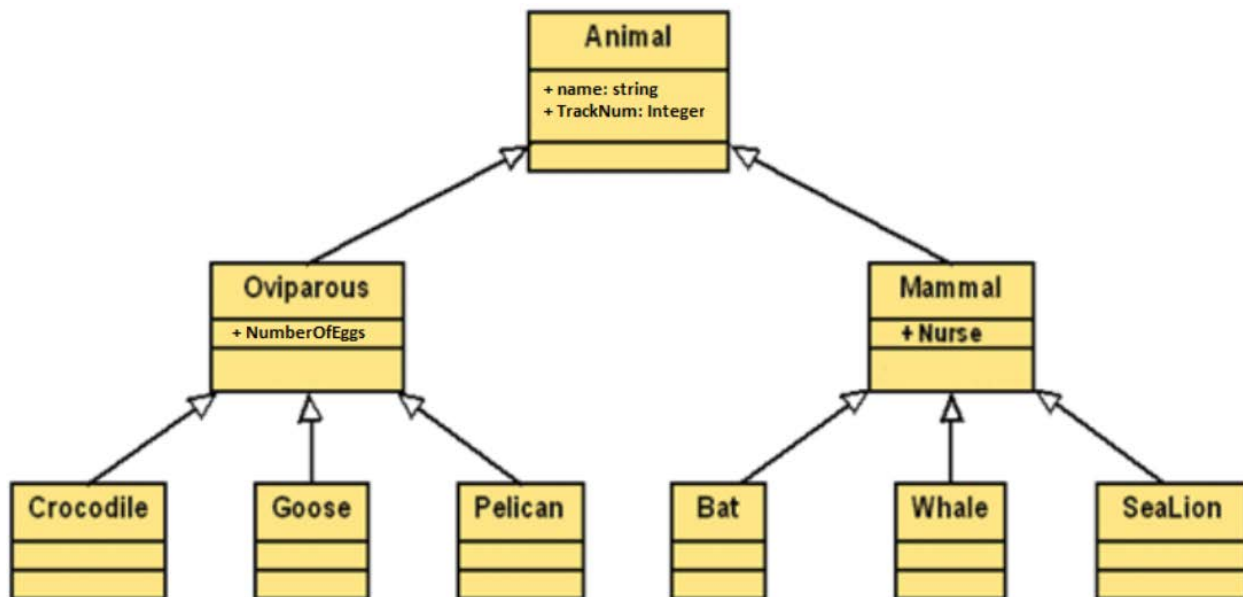
You will write a multi-class C++ interfacing prototype system for the Wildlife Zoo that incorporates Java for certain functionalities. The **functional requirements** for all components of the interface are as follows.

1. A **user menu** which displays the following options:
 - Load Animal Data
 - Generate Data
 - Display Animal Data
 - Add Record
 - Delete Record
 - Save Animal Data
2. **Generate Data:** Your supervisor has created this code for you in Java. In the C++ class you have been given, the createZooFile() function has been created for you. **You should not modify the Java code.**
 - a. For this function, you should make sure the user menu calls the createZooFile() function when the user selects “Generate Data” from the menu.
 - b. You will need to call this function and generate a file. When you are prompted for input, remember the following character limits:
 - i. Track #: 6 characters
 - ii. Name: 15 characters
 - iii. Type: 15 characters
 - iv. Sub-type: 15 characters
 - v. Eggs: Integer
 - vi. Nurse: Integer
3. **Load Animal Data:**
 - a. In your zipped folders, you have been given a Java class that generates a fixed length file. It will prompt the user to enter data for the following fields:
 - Track #: Received from the RFID chip technology
 - Name: Name given to the animal
 - Type: Oviparous (egg laying) or Mammal (primarily non-egg laying)
 - Sub-type: Crocodile, Bat, Whale, and so on
 - Eggs: Number of eggs laid
 - Nurse: Should read 0 if the animal is not nursing, 1 if it is
 - b. Data in the file will look like this. Column widths in the actual file will be fixed and are respectively 6 characters, 15 characters, 15 characters, 15 characters, and two integer columns.



Track #	Name	Type	Sub-type	Eggs	Nurse
000001	Tick-Tock	Oviparous	Crocodile	2	0
000002	Fidget	Mammal	Bat	0	1
000003	Willy	Mammal	Whale	0	0
000004	Bailey	Mammal	Whale	0	1
000005	Goose Lee	Oviparous	Goose	0	0
000006	Honker	Oviparous	Goose	1	0
000007	Becky	Oviparous	Pelican	1	0
000008	Nigel	Oviparous	Pelican	0	0
000009	Fluke	Mammal	SeaLion	0	0
000010	Rudder	Mammal	SeaLion	0	1
000011	Bartok	Mammal	Bat	0	1

- c. You must give users the ability to load this data into memory via a vector within your system. Use the following UML class diagram to build your class hierarchy and inheritance. Notice the variables listed in the Animal, Oviparous, and Mammal classes. A text version is available on the last page of this document.



- d. The message "Load complete." will display after the data loads successfully.
4. **Display Animal Data:** Display data on the user's computer screen in tabular format (presented in the form of a table with rows and columns).



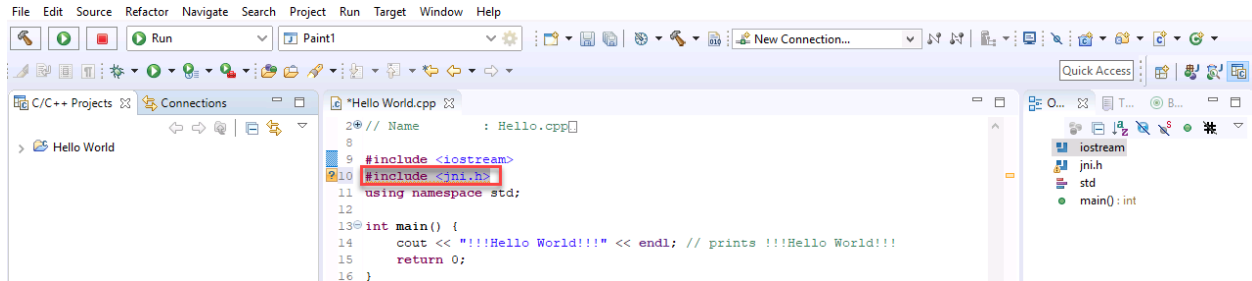
5. **Add Record:** Provide the capability to add new animal records.
 - a. When this menu option is selected, the application should prompt the user to enter values for the variables "Track#," "Name," "Type," "Sub-type," "Eggs," and "Nurse."
 - i. Make sure to validate input.
 - b. The user should have the ability to save or cancel the addition.
6. **Delete Record:** Provide the capability to delete animal records.
 - a. Delete an animal record by "Track#" when the user selects the "Delete Record" option from the menu.
 - i. Make sure to validate input.
 - b. The user should be prompted to confirm the deletion before the record is removed from the system.
 - c. The message "Animal successfully deleted" should display once the operation finishes.
7. **Save Data:** Permit the user to save modified animal data back to the input file, erasing the data that was previously there. The message "Save successfully completed" should display once finished.



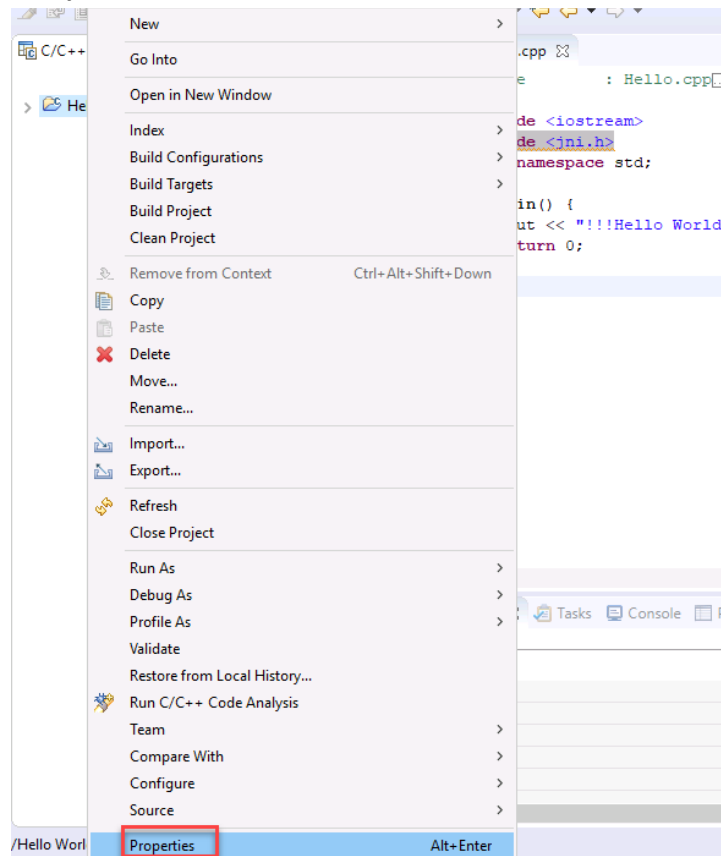
Setting Up JNI

Note: If you are not getting any errors on the `#include jni.h` line, you do not need to follow these steps.

To be able to properly link your C++ code to java, you need to load the Java Virtual Machine module. To do this, at the top of the code, type `#include <jni.h>`. At first, this will give you an error because Eclipse cannot “see” `jni.h`. You will fix this in the next step.

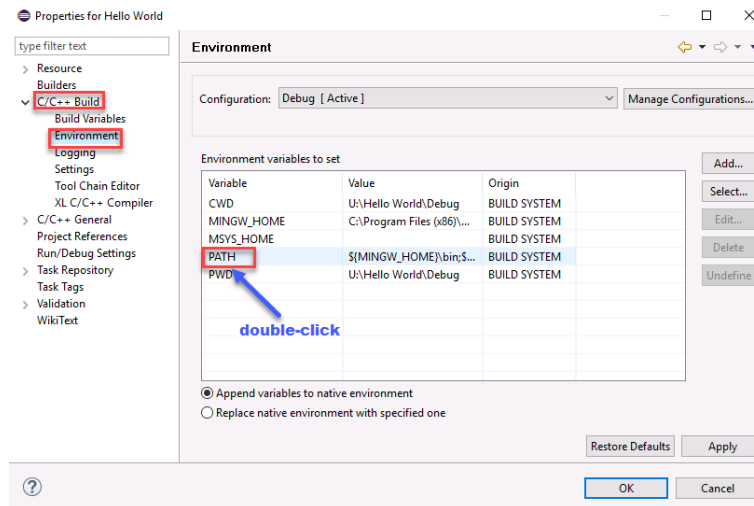


To get Eclipse to “see” `jni.h`, you need to build the path. In the Project Explorer window, right-click your C++ project, and select **Properties**.



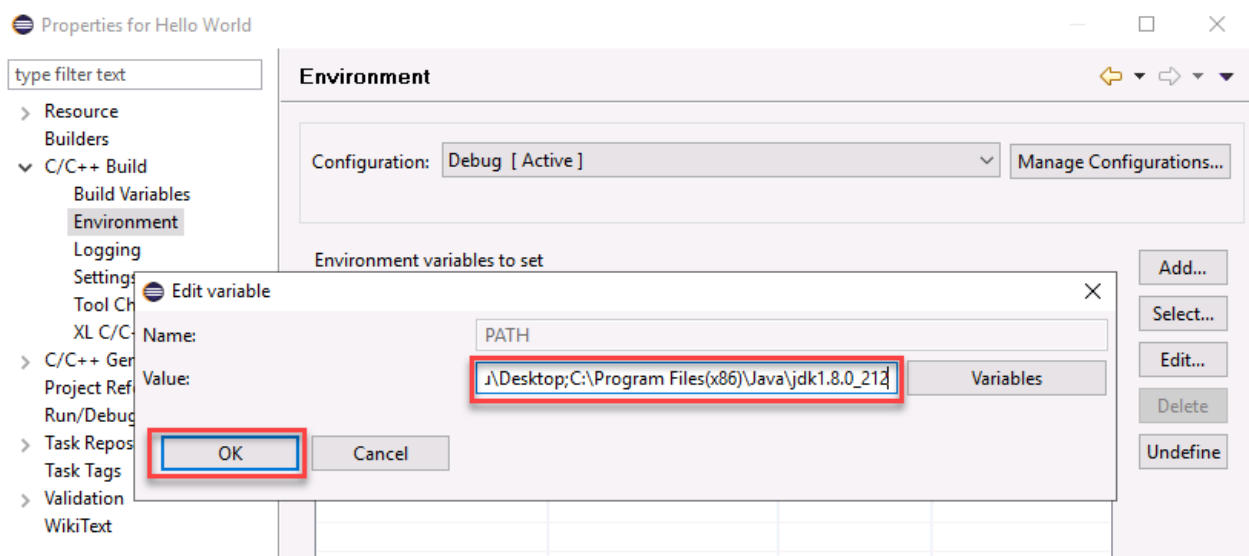


From the menu pane on the left, select **C/C++ Build**, then **Environment**. Under C/C++ builder, click on **Environment**. In the window, you will see the Environment pane, in the box “Environment variables to set”, double-click on the word **PATH**.




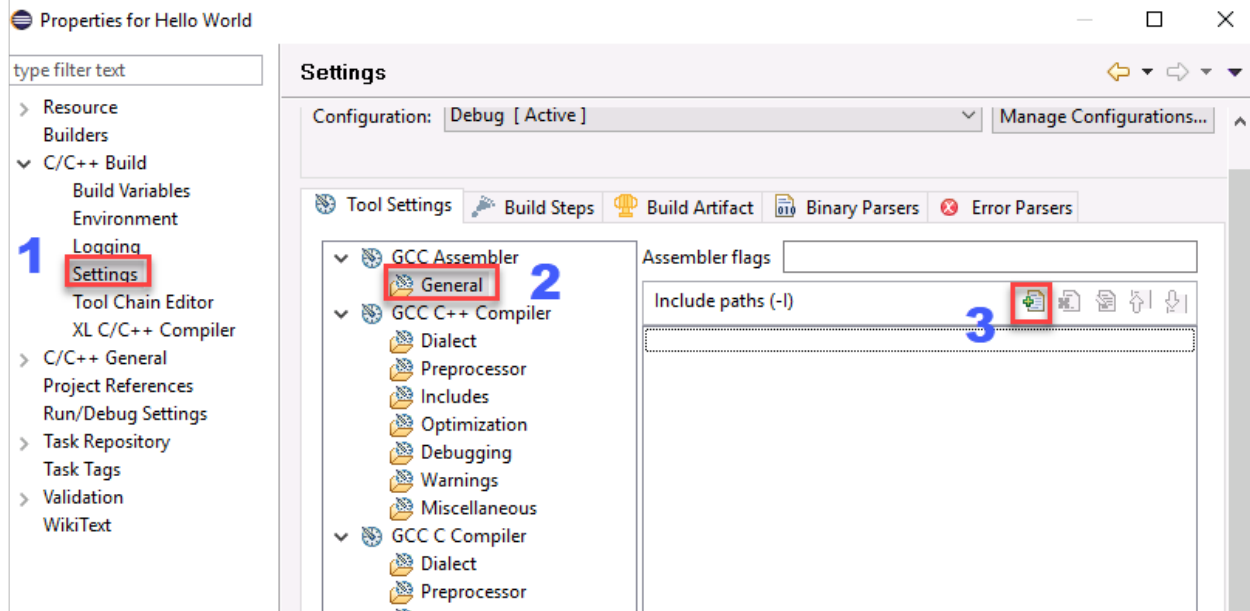
In the **Edit variable** dialog box, look at the box that says **Value**. There will already be some text in there. Click on the box and use your cursor to scroll to the very end of the text. Then type the following, including the semi-colon ;C:\Program Files (x86)\Java\jdk1.8.0_212. Then click **OK**.

NOTE: You cannot copy straight from this Word file on your computer into Eclipse, which is in Apporto. Instead, upload this Word document into Apporto. Then you will be able to copy the code into Eclipse.





In the menu pane, click on **Settings**, which is also under **C/C++ Build**. In the Settings pane, select **General** under **GCC Assembler**. Click the **Add...** icon 



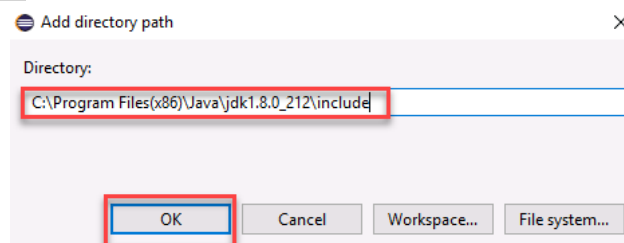
The **Add directory path** dialog box will pop up. You will need to add the following paths, one at a time. Type in each path and then click “OK.”

C:\Program Files (x86)\Java\jdk1.8.0_212\include

C:\Program Files (x86)\Java\jdk1.8.0_212\include\win32

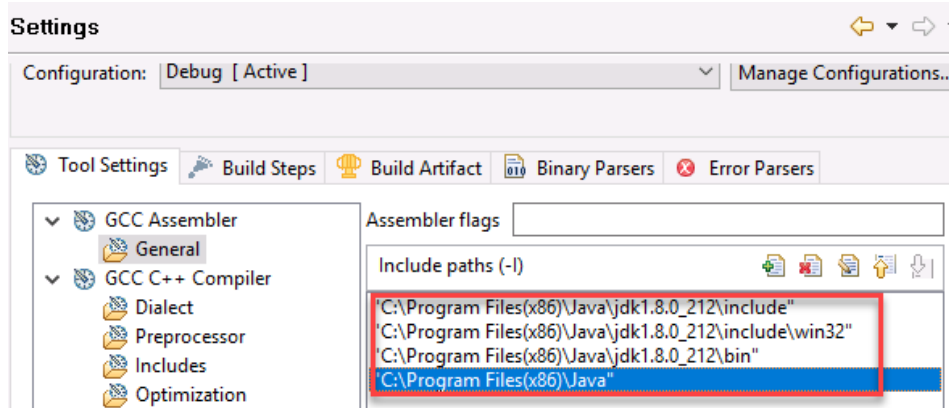
C:\Program Files (x86)\Java\jdk1.8.0_212\bin

C:\Program Files (x86)\Java





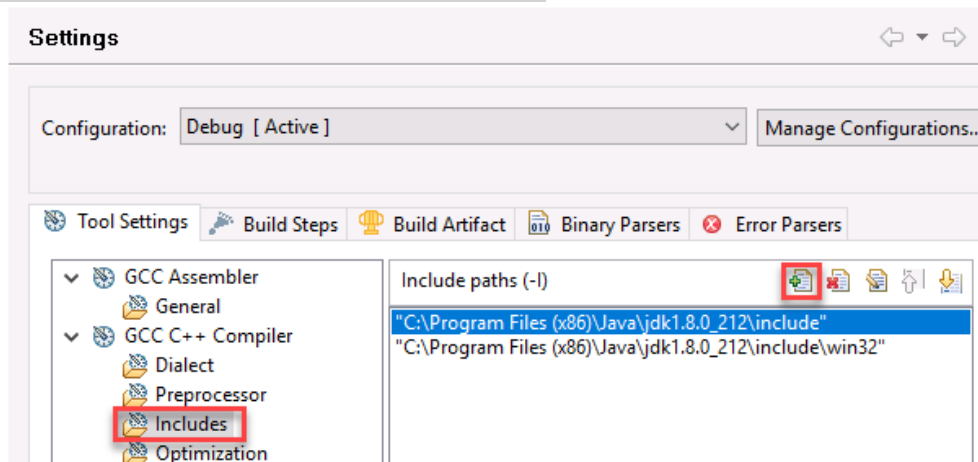
When you are finished, you should see all four paths in the window:



Next, in the **Settings** pane, select **Includes**, which is located under the **GCC C++ Compiler**. Then select the **Add...** button. Add the following paths the same way you added paths before.

C:\Program Files (x86)\Java\jdk1.8.0_212\include

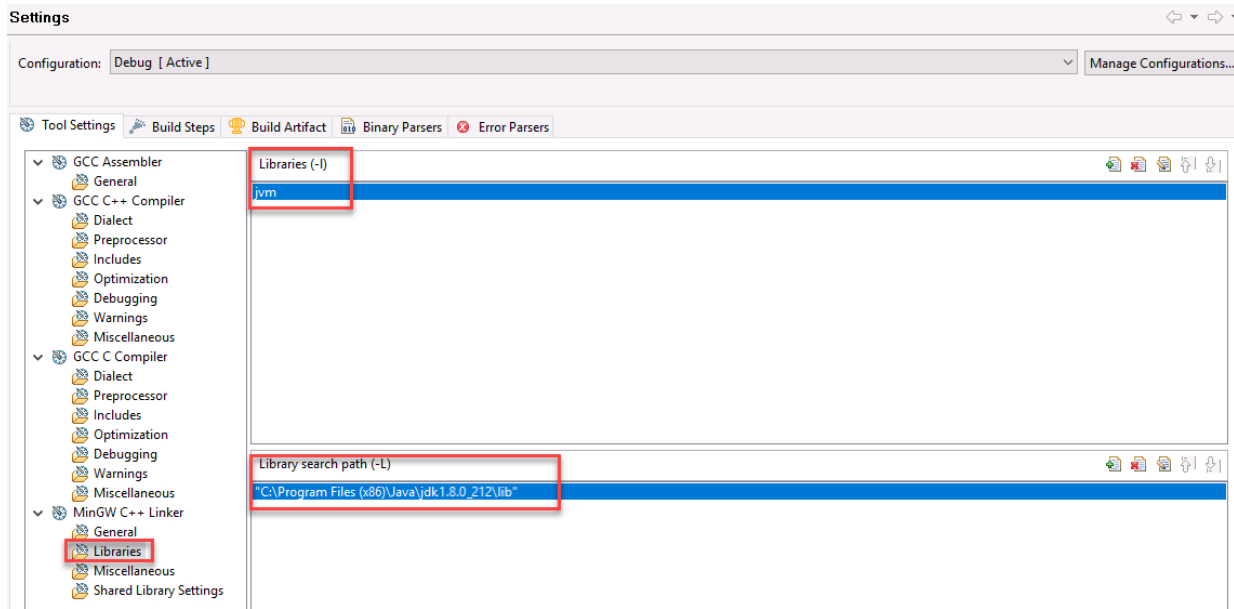
C:\Program Files (x86)\Java\jdk1.8.0_212\include\win32



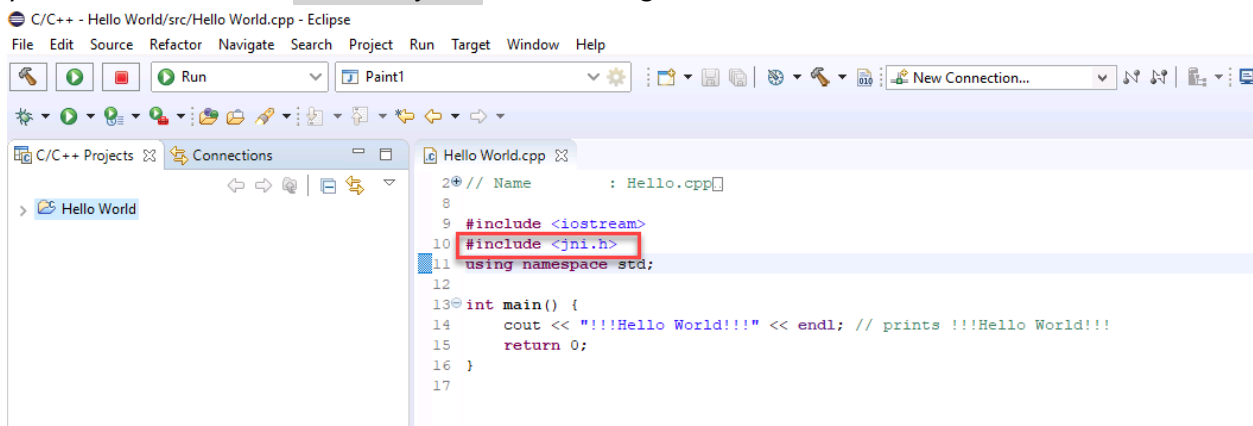


Finally, under **MinGW C++ Linker**, click **Libraries**. You will see two windows. The first is **Libraries**. Use the “Add...” button to add **jvm**. In the **Library search path (-L)** add the following path:

C:\Program Files (x86)\Java\jdk1.8.0_212\lib



Finally, click the “**OK**” button to close the windows. You may see a screen that prompts you to rebuild the libraries, click OK. When this is completed, you will be taken back to the Eclipse homescreen, and you should see that the **#include <jni.h>** error is now gone!





UML Class Diagram Text Version

In this text version, each class is created as a table with the class name as a header. There are arrows demonstrating inheritance connecting the different tables, each with alt-text descriptions.

The UML class diagram displays a total of nine classes. The abstract class is Animal, which has two variables: a String variable called "name" and an Integer called "code". Inheriting from that class are two classes: Oviparous, which has a variable +NumberOfEggs, and Mammal, which has a variable +Nurse. There are three classes which inherit from Oviparous: Crocodile, Goose, and Pelican. There are three classes which inherit from Mammal: Bat, Whale, and SeaLion.

