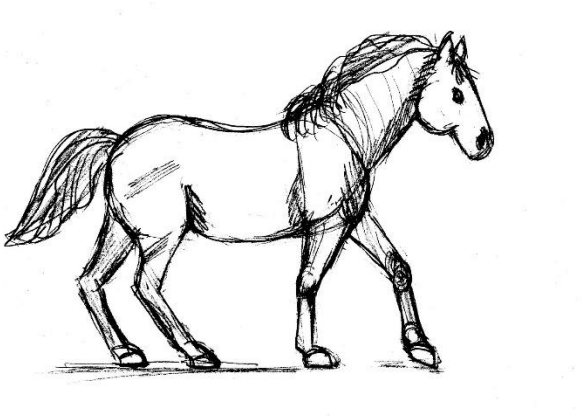
Java Programming – Creating a Class



Learning how to create and use classes and objects in Java is one of the most important concepts you will learn this semester. This is a KEY chapter on learning how to program in an object-oriented language. This assignment is designed to assess your ability to create a new class using a predefined UML diagram. *If needed, go back and review the format of a UML diagram in the textbook.*

A "driver" program that "uses" the class you create is supplied for this assignment. In a real-world setting, some programmers only use classes that have been predefined by others and they simply write driver programs. Other programmers create new classes for others to use. And still other programmers do both.

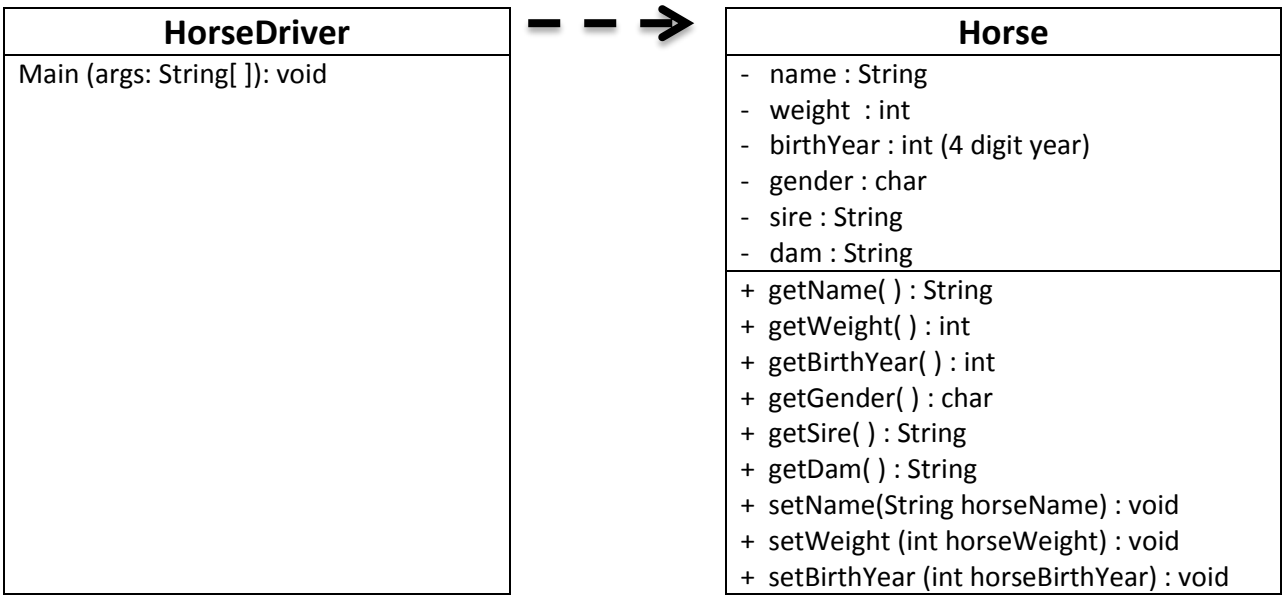
You will start this week by ONLY creating a new class. The driver is provided for you to test and use your new class. If your class is created per the UML diagram below, the driver will compile and run correctly.

Your assignment this week is to create a class called **Horse**. A driver program called **HorseDriver** is provided for this problem.

The **Horse** class contains instance data for a single horse:

- The name of the horse (a String)
- The weight of the horse (an integer)
- The 4-digit year the horse was born (an integer)
- The gender of the horse (M or F) (a char)
- The name of the sire (father) of the horse (a String)
- The name of the dam (mother) of the horse (a String)

The UML class diagram for **Horse** is below:



		+ setGender (char horseGender) : void + setSire(String horseSire) : void + setDam(String horseDam) : void + getAge( ) : int + toString( ) : String

(+ represents the public modifier and – represents the private modifier)

Steps to complete:

- ✓ Create the **Horse** class using the UML diagram on the previous page. *Use the name of the methods which are illustrated in the UML diagram so that the driver program runs correctly.*

- ✓ Create 2 constructors for the **Horse** class (each constructor will be named **Horse**):
  1. A **default constructor** with no parameters. The coding for the default constructor should set
     - **String** values to a space (" " – a space between the double quotes) OR to the empty string ("" -- no space between the double quotes). Either will work for our example.
     - **char** values to a space (' '-- a space between the single quotes) OR an empty char literal using one of the following:
       - '\u0000'
       - '\0'
       - (char) 0
     - **Numeric** values to zero.
  2. A **constructor** with 6 parameters (a value passed in for name, weight, birthYear, gender, sire, and dam – in this order). Coding should use the parameters to update the instance data items.

- ✓ Create each of the **accessors (getters) and mutators (setters)** which are illustrated in the UML diagram.

- ✓ Create a helping method named **getAge()** that will subtract the current year from the year the horse was born to calculate the age of the horse. The **getAge()** method should return that calculated age. There is an easy way for you extract the current year from your computer's date (do not input the current date).

  There is a **Calendar class** in the java.util package. To extract the current year from your computer's date, use the method below. This method returns an integer value so you can assign that value to an integer variable.

  Calendar.getInstance().get(Calendar.YEAR)

  To use the method above, include the following import statement in your class:

  import java.util.Calendar;

- ✓ Create a **toString( )** method which displays all class instance data in an easy to read format. The toString() should build a string of nicely formatted data from the class. It should not display any data. The driver program will decide what to do with the string.

It is critical that you use the names listed in the UML diagram. The driver program that was given to you to work with your Horse class expects the names listed in the UML diagram. You are not to make modifications to the driver program.

To test your **Horse** class:

- Save the **HorseDriver.java** file in the same folder where your **Horse.java** class was saved.
- Compile the **Horse** class and debug.
- Compile the **HorseDriver** class. Errors found at this point in the driver will be related to your **Horse** class and NOT the **HorseDriver** class. Debug **Horse**.
- When **Horse** and **HorseDriver** both compile cleanly, execute **HorseDriver**.

This will take a while to complete if this is your first exposure to object-oriented programming.

Zip your **Horse.java** and **HorseDrive.java** files together and submit the zip file in Blackboard for grading.

START EARLY – EMAIL YOUR INSTRUCTOR IF YOU HAVE QUESTIONS – LISTEN TO THE PODCAST IN BLACKBOARD.