Java programming II - Online

OOP

Polymorphism

Assignment 1 – Animal Farm

Mandatory

Farid Naisan
University Lecturer
Malmö University, Malmö Sweden
UCSD Extension, CA, USA

Table of Contents

The Animal Park 3
1 Objectives 3
2 Description 3
3 Specification 3
4 Class Diagram 4

Quality Standards and Guidelines

Requirements:

- ✓ You can use a standard text editor such as Windows Notepad.exe to write the source code, compile and run from the command line, or use other development environments (known as IDE, Integrated Development Environment) such as NetBeans, Eclipse, etc. to create the projects, write code, build and run from the IDE.
- ✓ All instance variables must be declared as private.
- ✓ Make your own assumptions whenever you find instructions unclear. Document your assumptions directly in the code or as an extra note to your instructor.
- ✓ In view of the fact that this and other assignments are designed for beginners, you may practice improvements, optimizations and enhancements to the coding structure and the user interface if you are not a beginner. However, certain instructions may be marked as mandatory in which case you are expected to follow the instructions.
- ✓ Write your name and the date on every file (as comments).
- ✓ Do not forget to document your source code by writing comments above or in front of statements.

Quality

- ✓ The application must have been compiled, tested and run satisfactorily before it is submitted. It is very important to maintain a good style and code structure and a normally functioning program, rather than presenting a well functioning program, but with a poor code quality. Higher grades are given to projects containing well-structured and reasonably documented code, and a satisfactorily working program.
- ✓ All identifiers (class names, variable names and method names) should be chosen carefully so that they express their purpose. The suggested solutions to the exercises can be taken as a guide. Short names that are not expressive (ex a, fnc, etc.) and should be strictly avoided.

Assessment:

- ✓ The assignment receives a letter grade A-F which can be then translated to a grade type of your choice. Projects that do not meet the minimum criteria for a passing grade will be returned for complementary work. The final result for all assignments will be determined as a weighed average of all assignments plus the instructor's judgment of the progress you make throughout the course.

Submission

- ✓ The files are to be packed into a ZIP or RAR file and submitted in the Blackboard. A trial version of WinRar can be downloaded from the Internet from www.winrar.com . Make sure to include all the files that are part of your project.
- ✓ Although it is allowed to discuss solutions and ideas with other classmates, the assignment is done and submitted individually.

The Animal Park

1 Objectives

You remember the three most important principles of OOP, capsulation, Inheritance and Polymorphism. This assignment will help you to refresh your memories of the basic course and your general skills in Java. In addition to basic java data structures and syntax, the assignment will give you training in polymorphism, by implementing interface, abstract classes and dynamic binding.

2 Description

A city zoo would like you to write a program for handling their animals and facilities. To begin with, they would like computerize the registration of the animals housed in the zoo. They categorize the animals according to what the animals eat:

PlantEater

- Horse
- Giraff
- Deer

MeatEater

- Wolf
- Lion
- Dog

A PlantEater eats

- Grass
- Leaves

A MeatEater eats::

- Pork
- Beef

Animals are live in:

- Cage
- Stable
- Outdoor
- Indoor

The details above are not very exact. You may add new ones or change the items.

3 Specification

3.1 The coding done in this part will be used in later assignments. Therefore, it is very essential to apply a good design and code with maintainability, extensibility and readability in mind. The achieve these goals, we will use as much of the OOP aspects as possible so that further development can be carried out easily..

3.2 The program in this version needs not to have any user interface, as it is intended to develop a graphical user interface in Assignment 2. The application will be tested for a number of hard-coded test values.

3.3 Create an abstract class **Animal** to encapsulate all the things that animals have in common. The following attributes should be implemented in the class.

3.3.1 Animal's name
3.3.2 ID number
3.3.3 Number of legs
3.3.4 Housing form (stable, cage, outdoor, indoor). Use an enum for this data.

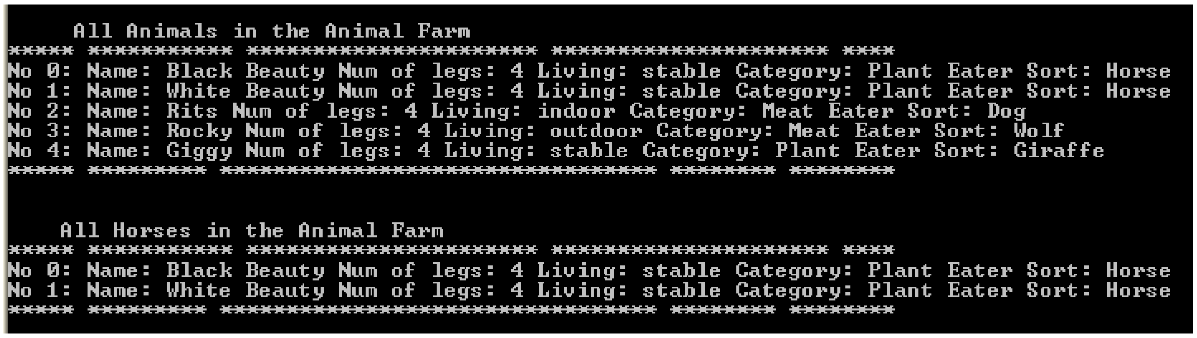
3.4 Now let's put some polymorphic rules by creating an Interface, **IAnimal**, for sub-classes to implement the following operations

3.4.1 A method **eat** that receives name of a food type as a String and returns back a String that either says "*Yummy*" or "*Uuusch*" (or similar text) depending on whether the animal likes or dislikes the food. To detect this, match the food (in parameter) with a number predefined food types in the program (classes **PlantEater**, **MeatEater**).

3.4.2 Setter functions for name, ID number, and number of legs for every animal.

3.5 No input/output operation is to be included in any class except the class **UserInerface** which is to be fully responsible for all communications with the user.

3.6 The application is to be tested for a number of hard-coded animal objects, that should include a couple of horses, a dog, a giraffe and a wolf, as shown in the run example below. This operation should be organized as a method (setTestValues()) in the AnimalManager class. The class **UserInterface** is to call this function to test the rest of the application.



```
     All Animals in the Animal Farm
***** *********** ********************** ********************* ****
No 0: Name: Black Beauty Num of legs: 4 Living: stable Category: Plant Eater Sort: Horse
No 1: Name: White Beauty Num of legs: 4 Living: stable Category: Plant Eater Sort: Horse
No 2: Name: Rits Num of legs: 4 Living: indoor Category: Meat Eater Sort: Dog
No 3: Name: Rocky Num of legs: 4 Living: outdoor Category: Meat Eater Sort: Wolf
No 4: Name: Giggy Num of legs: 4 Living: stable Category: Plant Eater Sort: Giraffe
***** ********* ********************************* ******** ********


    All Horses in the Animal Farm
***** *********** ********************** ********************* ****
No 0: Name: Black Beauty Num of legs: 4 Living: stable Category: Plant Eater Sort: Horse
No 1: Name: White Beauty Num of legs: 4 Living: stable Category: Plant Eater Sort: Horse
***** ********* ********************************* ******** ********
```

4 Class Diagram

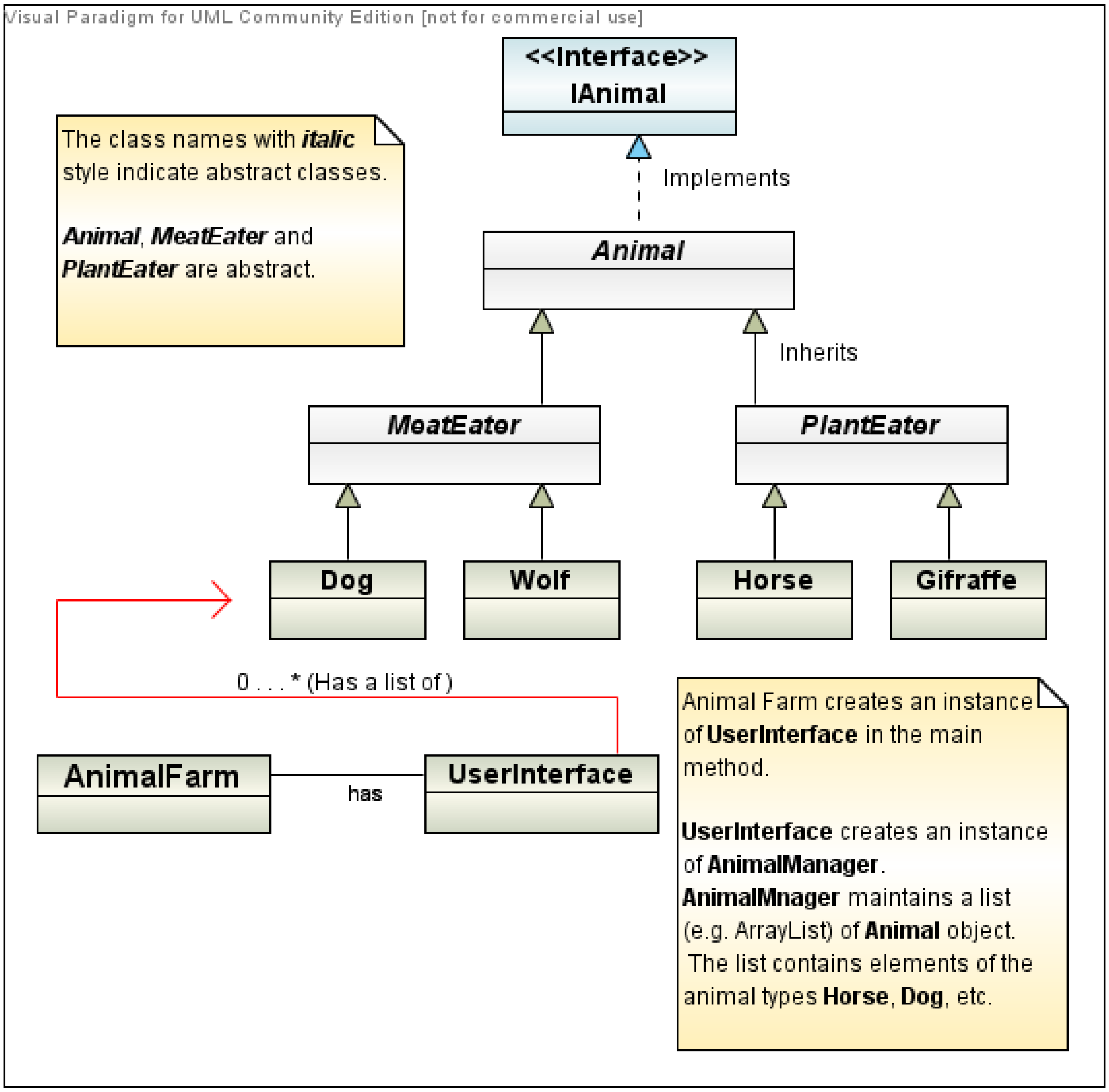
The class diagram presented here illustrates a recommended solution. The application can be designed in many ways, but this design fulfills the objectives of this assignment very well as you get the chance to use all your OOP skills.

The start class can look like this:

```
public class AnimalFarm
{
    /**
     * The main method start the application by calling
     * the start-method of the UserInterface object
     */
    public static void main(String[] args)
    {
        UserInterface prog = new UserInterface();
        prog.start();

    }

}
```

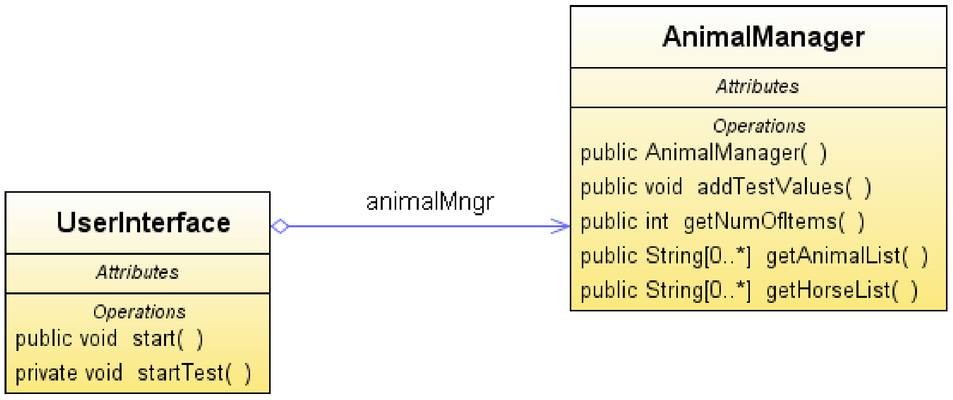

Visual Paradigm for UML Community Edition [not for commercial use]
<<Interface>> IAnimal
Implements
Animal
Inherits
MeatEater
PlantEater
Dog
Wolf
Horse
Gifraffe
0 . . . * (Has a list of)
AnimalFarm
has
UserInterface
The class names with italic style indicate abstract classes.
Animal, MeatEater and PlantEater are abstract.
Animal Farm creates an instance of UserInterface in the main method.
UserInterface creates an instance of AnimalManager.
AnimalMnager maintains a list (e.g. ArrayList) of Animal object. The list contains elements of the animal types Horse, Dog, etc.

The attributes and methods of every class are given below (without any particular order). You don't have to following these step by step but make sure that you apply better techniques.

NOTE: The class **AnimalManager** can make use of an ArrayList

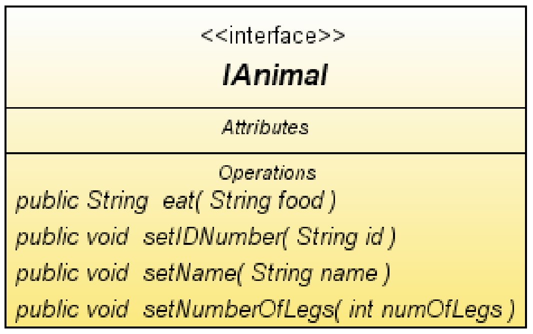
```
private List<Animal> animals;
```

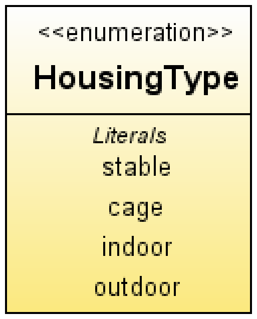
Use the following link to read more about ArrayList:
http://www.javadeveloper.co.in/java-example/java-arraylist-example.htm

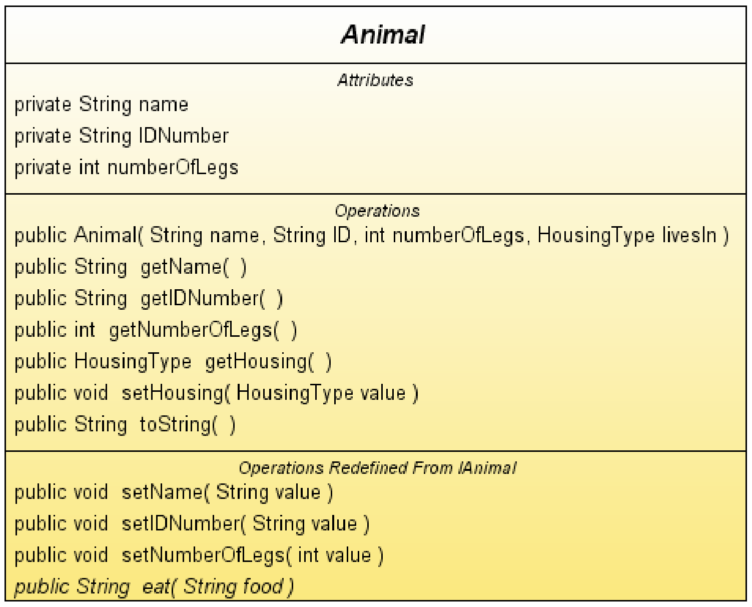

UserInterface
Attributes
Operations
public void start( )
private void startTest( )
animalMngr
AnimalManager
Attributes
Operations
public AnimalManager( )
public void addTestValues( )
public int getNumOfItems( )
public String[0..*] getAnimalList( )
public String[0..*] getHorseList( )

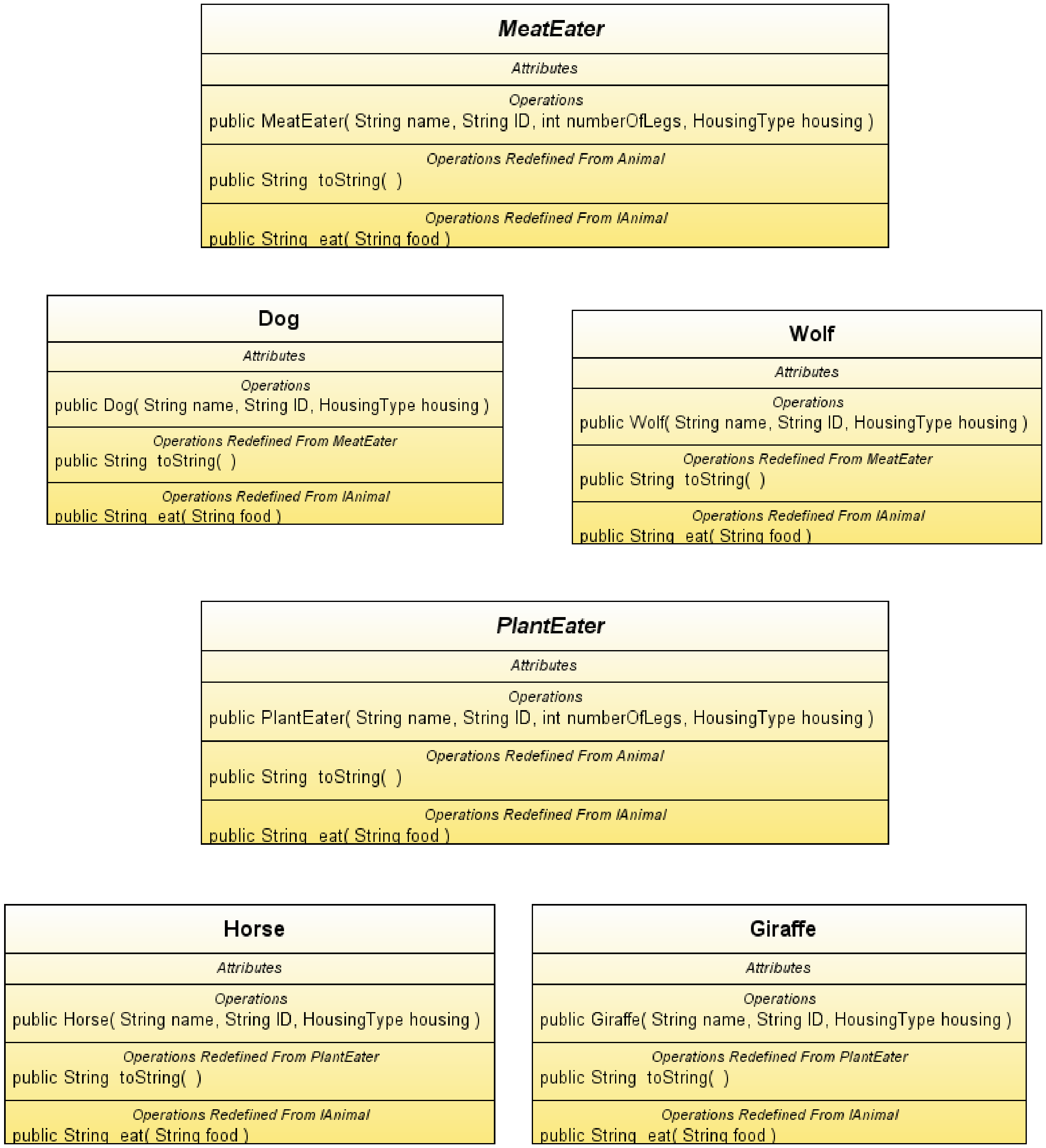
For defining housing types, you may use the structure given here:
Find out more about enums at:
http://java.sun.com/docs/books/tutorial/java/javaOO/enum.html

```
public enum HousingType
{
      stable,
      cage,
      indoor,
      outdoor
}
```


<<interface>>
IAnimal
Attributes
Operations
public String eat(String food)
public void setIDNumber(String id)
public void setName(String name)
public void setNumberOfLegs(int numOfLegs)




Animal
Attributes
private String name
private String IDNumber
private int numberOfLegs
Operations
public Animal(String name, String ID, int numberOfLegs, HousingType livesIn)
public String getName( )
public String getIDNumber( )
public int getNumberOfLegs( )
public HousingType getHousing( )
public void setHousing(HousingType value)
public String toString( )
Operations Redefined From IAnimal
public void setName(String value)
public void setIDNumber(String value)
public void setNumberOfLegs(int value)
public String eat(String food)



MeatEater

Attributes

Operations
public MeatEater(String name, String ID, int numberOfLegs, HousingType housing)

Operations Redefined From Animal
public String toString( )

Operations Redefined From IAnimal
public String eat(String food)

Dog

Attributes

Operations
public Dog(String name, String ID, HousingType housing)

Operations Redefined From MeatEater
public String toString( )

Operations Redefined From IAnimal
public String eat(String food)

Wolf

Attributes

Operations
public Wolf(String name, String ID, HousingType housing)

Operations Redefined From MeatEater
public String toString( )

Operations Redefined From IAnimal
public String eat(String food)

PlantEater

Attributes

Operations
public PlantEater(String name, String ID, int numberOfLegs, HousingType housing)

Operations Redefined From Animal
public String toString( )

Operations Redefined From IAnimal
public String eat(String food)

Horse

Attributes

Operations
public Horse(String name, String ID, HousingType housing)

Operations Redefined From PlantEater
public String toString( )

Operations Redefined From IAnimal
public String eat(String food)

Giraffe

Attributes

Operations
public Giraffe(String name, String ID, HousingType housing)

Operations Redefined From PlantEater
public String toString( )

Operations Redefined From IAnimal
public String eat(String food)

Please use the forum for this module to ask questions about things that are unclear.

Good Luck!

Programming is fun. Never give up. Ask for help!

Farid Naisan, Instructor