T5 - Java Seminar

T-JAV-500

Day 02

Object Oriented Programming

3.3

Day 02

language: Java

- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.

As of today, you're going to learn Object Oriented Programming (OOP).
In some previous exercises you may have briefly used it, but from now on you will use it every day.
To help you with your research, here are today's concepts (in no particular order):

- Classes definition
- Objects and instantiations
- Constructors
- Methods and attributes
- New operator
- Methods and attributes visibility
- Optional parameters

EXERCISE 01

File to hand in: ./ex_01/Gecko.java

Create a new class "Gecko".
Every time a new Gecko is created, it will automatically display "Hello!" followed by a newline.

Here is a test example:

```
public class Example {
    public static void main(String[] args) {
        Gecko arthur = new Gecko();
        Gecko benjy = new Gecko();
     }
  }
```

```
Terminal
~/T-JAV-500> java Example
Hello!
Hello!
```

EXERCISE 02

File to hand in: ./ex_02/Gecko.java

Copy your "Gecko.java" from exercise 01.
Add a new constructor so that it is possible to pass a name as parameter for the newly-created Gecko.
If a name is specified, "`Hello <Name>!`" followed by a newline must be displayed.
This parameter will need to be stored in a public "name" attribute.

If no names are specified, the previous constructor is called, but you must set the "name" attribute to the value "Unknown".
Here is a test example:

```
public class Example {
    public static void main(String[] args) {
        Gecko arthur = new Gecko("Arthur");
        Gecko benjy = new Gecko();

        System.out.println(arthur.name);
        System.out.println(benjy.name);
    }
}
```

```
Terminal
~/T-JAV-500> java Example | cat -e
Hello Arthur!$
Hello!$
Arthur$
Unknown$
```

EXERCISE 03

File to hand in: ./ex_03/Gecko.java

Once again, copy your "Gecko.java" from the previous exercise.

Ah, speaking of names…
From now on, the "name" attribute should not be public.

You have to find out for yourself what it should be.

However, so that the name is available outside the object, you need to create your very first method! It must be called **getName** and return… the Gecko's name!

Here is a test example:

```
public class Example {
    public static void main(String[] args) {
        Gecko arthur = new Gecko("Arthur");
        Gecko benjy = new Gecko();

        System.out.println(arthur.getName());
        System.out.println(benjy.getName());
    }
}
```

```
Terminal
~/T-JAV-500> java Example | cat -e
Hello Arthur!$
Hello!$
Arthur$
Unknown$
```

EXERCISE 04

File to hand in: ./ex_04/Gecko.java

Copy your "Gecko.java" from the previous exercise.

Add a new "age" attribute to your Gecko class.
It should be possible to set it as a second parameter during the construction of the object.

Previous constructor rules still apply.

This attribute must have its own getter and setter, respectively **getAge** and **setAge**.

Also add a new "status" method to your Gecko.
This method must take no parameters and display a sentence according to the Gecko's age.

You must use a "switch" statement and you are not allowed to use the "if" keyword in this method.

The method must display the following sentences:

- "Unborn Gecko", if the age is 0.
- "Baby Gecko", if the age is 1 or 2.
- "Adult Gecko", if the age is between 3 and 10.
- "Old Gecko", if the age is between 11 and 13.
- "Impossible Gecko", otherwise.

Each of these sentences must be followed by a newline.

EXERCISE 05

File to hand in: ./ex_05/Gecko.java

It is time to give the gift of gab to our Geckos!
First, copy your previous "Gecko.java".
Add a new public method to it, called **hello**.

When called with a string, it must display "`Hello <string>, I'm <Name>!`",
with `<string>` being the string given as parameter
and `<Name>` being the name of the Gecko.
However, if an integer is given as parameter, it must display "`Hello, I'm <Name>!`" as often as the number given as parameter.

Every messages must be followed by a newline.

In all other cases, the method does nothing.

```
public class Example {
    public static void main(String[] args) {
        Gecko mimi = new Gecko("mimi");
        mimi.hello("Titi");
        mimi.hello(2);
     }
  }
```

Terminal

```
~/T-JAV-500> java Example | cat -e
Hello mimi!$
Hello Titi, I'm mimi!$
Hello, I'm mimi!$
Hello, I'm mimi!$
```

EXERCISE 06

File to hand in: ./ex_06/Gecko.java

Copy your "Gecko.java" from the previous exercise.
Add a new **eat** method to your Gecko. It takes a string as parameter and returns nothing.
If the value of the string given as parameter is equal to "Meat", the Gecko must display:

```
Yummy!
```

If the value is equal to "Vegetable", your Gecko must say:

```
Erk!
```

If the value is anything else, the Gecko must display:

```
I can't eat this!
```

As usual, every sentence will be followed by a newline.

The 'eat' method must be case insensitive!

Moreover, add a new "energy" attribute to our Gecko.
By default it is equal to 100.

Add a setter and a getter for this attribute (**getEnergy** and **setEnergy**).
Every time our Gecko eats something, he will win or lose some energy.

If he eats meat, he will win 10 Energy.
However, if he eats vegetables, he will lose 10 Energy (a Gecko is carnivorous).
In all other cases his energy will not be modified.

A gecko's energy should always be between 0 and 100 (included).

EXERCISE 07

File to hand in: ./ex_07/Gecko.java

Copy your “Gecko.java” from the previous exercise.
Implement a new **work** method in our Gecko class that takes no parameter and returns nothing.

With every call, if the Gecko has at least 25 energy, it will display:

```
I'm working T.T
```

Then the energy will decrease by 9 (he is working 8 hours a day so he needs enough energy to work the whole day).

If the method is called with less than 25 energy, it will display:

```
Heyyy I'm too sleepy, better take a nap!
```

Then it will give your Gecko back 50 energy.

As usual, every sentence will be followed by a newline.

EXERCISE 08

Files to hand in: ./ex_08/Gecko.java, ./ex_08/Snake.java

Let's implement a new **fraternize** method that takes one parameter.
If the parameter is a Gecko Object, our Gecko will be happy and go drink with his friend.
It will cost both of them 30 Energy and they will both say (starting with the current Gecko):

```
I'm going to drink with <otherName>!
```

If one of them doesn't have enough energy, he will say:

```
Sorry <otherName>, I'm too tired to go out tonight.
```

and the other will answer:

```
Oh! That's too bad, another time then!
```

If both of them are too tired to go out, they will both say:

```
Not today!
```

If the parameter is a Snake, and if our energy is greater than or equal to 10, it must be set to 0 and we will say:

```
LET'S RUN AWAY!!!
```

If our energy is less than 10, we will play dead and display:

```
...
```

Feel free to add any method that you reckon necessary.