



T5 - Java Seminar

T-JAV-500

Day 03

Packages



Day 03

language: Java



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

Let's keep going deeper into OOP!

You will today keep using all yesterday's concepts, and you will also discover a few new things:

- static keyword
- packages



Unless otherwise specified, all messages must be followed by a newline.



EXERCISE 01

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

Create a new class, named `Mars`, that has an `id` attribute, a getter (`getId`), but no setter.

You must create your class so that the `id` of the first instance of `Mars` is 0, the `id` of the second instance is 1...



static?!

```
public class Example {  
    public static void main(String[] args) {  
        Mars rocks = new Mars();  
        Mars lite  = new Mars();  
        Mars snack = new Mars();  
  
        System.out.println(rocks.getId());  
        System.out.println(lite.getId());  
        System.out.println(snack.getId());  
    }  
}
```

```
Terminal  
~/T-JAV-500> java Example  
0  
1  
2
```



Starting at the following exercise, the files and classes will be reused and expanded across exercises.

EXERCISE 02

File to hand in: ./Astronaut.java

Create a new “Astronaut” class, with the following attributes:

- **name:** a string describing the name of the Astronaut,
- **snacks:** an integer describing the number of snacks our Astronaut possess,
- **destination:** a string describing the destination of the Astronaut,
- **id:** an integer describing the id of the Astronaut.

The name must be passed during the creation of the Astronaut.



It is mandatory.

His snack will be initialized to 0 and his destination to null.

The id must be unique and be incremented for each new Astronaut being created (i.e the first Astronaut's id must be 0, then the second one's 1,...).

Also, every Astronaut being created must display:

```
[name] ready for launch!
```

where [name] is the name of the Astronaut.

All these attributes must have an associated getter but no setter.

```
public class Example {  
    public static void main(String[] args) {  
        Astronaut mutta = new Astronaut("Mutta");  
        Astronaut hibito = new Astronaut("Hibito");  
  
        System.out.println(mutta.getId());  
        System.out.println(hibito.getId());  
    }  
}
```



```
Terminal
~/T-JAV-500> java Example
Mutta ready for launch!
Hibito ready for launch!
0
1
```

EXERCISE 03

Files to hand in: ./chocolate/Mars.java
./planet/Mars.java

Copy your `Mars` class from the first exercise, without changing it.

Create another `Mars` class representing the planet.

In order to differentiate between the two `Mars`, put the first one (the one from exercise 1) in a package called “chocolate” and the second one in a package called “planet”.

Add an attribute **landingSite** of type `String` to the planet and his getter. When creating a instance of the planet `Mars`, you must specify the name of the landing site in the constructor.

```
import chocolate.*;
import planet.*;

public class Example {
    public static void main(String[] args) {
        chocolate.Mars snack = new chocolate.Mars();
        planet.Mars rock = new planet.Mars("Viking 1");

        System.out.println(snack.getId());
        System.out.println(rock.getLandingSite());
    }
}
```

```
Terminal
~/T-JAV-500> java Example
0
Viking 1
```



EXERCISE 04

File to hand in: ./Astronaut.java
 ./chocolate/Mars.java
 ./planet/Mars.java



In this exercise, every time we ask for **[Name]** to be displayed, it should be replaced by the name of the Astronaut.

For example **[Name]: Nothing to do.**

becomes (for an Astronaut named "Mutta") **Mutta: Nothing to do.**

It is time for our Astronaut to start working!

Create a new method **doActions** taking an optional parameter.

If no parameter is given the method simply displays:

```
[Name]: Nothing to do.
```

If the parameter is a "planet.Mars", display:

```
[Name]: Started a mission!
```

You will also need to store the planet landing site as our Astronaut's new destination.

If a "chocolate.Mars" is given, display:

```
[Name]: Thanks for this mars number [Mars id]
```

where **[Mars id]** should obviously be replaced by the Mars id...

Also, if our Astronaut received a snack, you will also need to increment his "snacks" attribute by one.

After each previous sentence, if the astronaut has no destination, it will also display:

```
[Name]: I may have done nothing, but I have [x] Mars to eat at least!
```

where **[x]** is the number of snacks possessed by our Astronaut.

EXERCISE 05

Files to hand in: ./planet/moon/Phobos.java
 ./planet/Mars.java

Create a **Phobos** class in the **Phobos.java** file. This class must be in a **moon** package, which is, itself, defined in the **planet** package.



Your `Phobos` class must have a private attribute named `mars` with a getter (**`getMars`**), but no setter. This attribute must be specified upon creation; it is a representation of a `planet.Mars` followed by a landing site (`landingSite`) (thus also stored in the class and having its own getter).

During its creation, if it correctly received a “`planet.Mars`”, it must display:

```
Phobos placed in orbit.
```

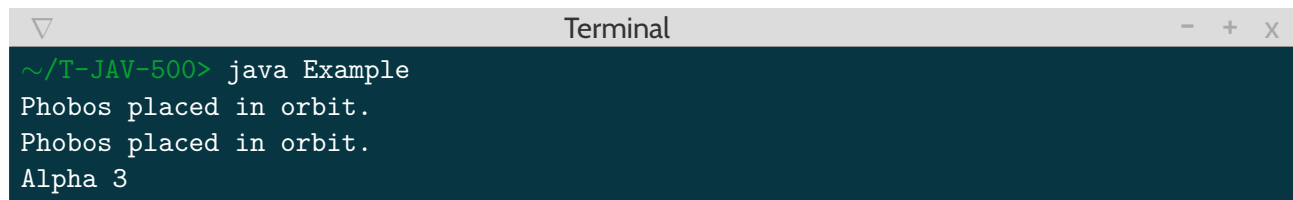
Otherwise, it must display:

```
No planet given.
```

```
import planet.*;

public class Example {
    public static void main(String[] args) {
        planet.Mars titi = new planet.Mars("Here and there");
        planet.Mars toto = new planet.Mars("Up");
        planet.moon.Phobos phobos1 = new planet.moon.Phobos(titi, "
            Alpha 3");

        new planet.moon.Phobos(toto, "Beta 1");
        System.out.println(phobos1.getLandingSite());
    }
}
```



```
~/T-JAV-500> java Example
Phobos placed in orbit.
Phobos placed in orbit.
Alpha 3
```

EXERCISE 06

File to hand in: `./planet/Mars.java`
`./Astronaut.java`
`./Team.java`

Create a new “`Team`” class that represents a team of astronaut. Its constructor must take the team name as parameter.

Create a getter (**`getName`**), but no setter for this attribute. Create a few methods to manipulate our team:

- **add**
It takes an `Astronaut` as parameter and add it to the team.



- **remove**

It takes an `Astronaut` as parameter and removes it from the team.

- **countMembers**

It returns the number of `Astronaut` currently on our team.

- **showMembers**

It displays the members that are on the team, the following way:

```
[Team name]: [Astronaut 1] on mission, [Astronaut 2] on standby.
```

Obviously, “on mission” is displayed if the `Astronaut` is currently on a mission.

Otherwise “on standby” is displayed.

If no member is in the team, don’t display anything.

```
import planet.*;

public class Example {
    public static void main(String[] args) {
        Astronaut mutta = new Astronaut("Mutta");
        Astronaut hibito = new Astronaut("Hibito");
        Astronaut serika = new Astronaut("Serika");

        Team spaceBro = new Team("SpaceBrothers");

        spaceBro.add(mutta);
        spaceBro.add(hibito);
        spaceBro.add(serika);

        System.out.println(spaceBro.countMembers());

        planet.Mars titi = new planet.Mars("Hill");
        mutta.doActions(titi);

        spaceBro.showMembers();
        spaceBro.remove(hibito);
        System.out.println(spaceBro.countMembers());
    }
}
```

```
Terminal
~/T-JAV-500> java Example
Mutta ready for launch!
Hibito ready for launch!
Serika ready for launch!
3
Mutta: Started a mission!
SpaceBrothers: Mutta on mission, Hibito on standby, Serika on standby.
2
```


EXERCISE 07

Files to hand in: ./planet/moon/Phobos.java
./chocolate/Mars.java
./planet/Mars.java
./Astronaut.java
./Team.java

Add a new method to your Team, **doActions**.

The goal of this method is to call all of the Team's Astronaut's **doActions** with the received parameter.
If no parameter is received, display:

```
[Team name]: Nothing to do.
```



Display it only once.



If `chocolate.Mars` is received as parameter, we will admit that the team share the chocolate but it still count as a full chocolate for each astronauts.

Now that our Astronauts have more experience, they can also go on a mission to Phobos.
You will need to modify your Astronaut class.