



# Work on project. Stage 6/6: Ĉu ĝi estas la kato?

Project: [Guess the Animal](#)

## Ĉu ĝi estas la kato?

 Hard  10 minutes  64 users solved this stage. Latest completion was **7 days ago**.

### Description

Your program is already really cool: as a matter of fact, it's too cool to be available only to the English speakers. In this stage, we will make our game international by adding support for different countries and languages.

As an example, we will use a neutral language [Esperanto](#). Resource files for this language are available for download in the attachment for this step. To pass the tests, you need to internationalize the program and add support for this language.

We also encourage you to add support for another language of your choice: it could be your native language or a language you are learning.

[Join a study group for the project Guess the Animal](#)

Discuss your current project with fellow learners and help each other.

### Theory

Program internationalization is a complex process. To be ready to implement it, check out an [article on localization on baeldung.com](#). To support different languages, Java uses [resource files](#), so you want to be confident working with them, too.

To add a resource file to the project, you need to create a subdirectory *main/resources* in the *src* directory: this is where we should put the resource files. They are typically text property files (.properties), but they can also be XML files or regular Java classes. If they are Java classes, then we can not only use text strings but also return Java objects specific to the selected language. Let's take a closer look at using Java classes.

To create a Java resource class, you must inherit from the class `ListResourceBundle`. We also need to rewrite the method `getContents` to return a two-dimensional array of objects. Here is an example of a class for the default language:

```
public class App extends ListResourceBundle {
    @Override
    protected Object[][] getContents() {
        return new Object[][]{
            {"hello", "Hello!"},
            {"bye", new String[]{
                "Bye!",
                "Bye, bye!",
                "See you later!",
                "See you soon!",
                "Have a nice one!"
            }},
            {"animal.name", (UnaryOperator<String>) name -> {
                If (name.matches("[aeiou].*") {
                    return "an " + name;
                } else {
                    return "a " + name;
                }
            }},
            {"animal.question", (UnaryOperator<String>) animal -> "Is it " + animal + "?"}
        };
    }
}
```

In this example, we have created an `App` resource class with three keys: *hello*, *bye*, and *animal.name*. The key “hello” is of the type `String`, the key “bye” is an array of strings, and the key “animal.name” is a `UnaryOperator` object.

Now you can load this class in your program and get resources using the following code:

```
var appResource = ResourceBundle.getBundle("App");

var helloString = appResource.getString("hello");
var byeStringArray = appResource.getStringArray("bye"));
var animalName = (UnaryOperator) appResource.getObject("animal.name");
```

For Esperanto, this class should be named *App\_eo*. It should look something like this:

```
public class App_eo extends ListResourceBundle {
    @Override
    protected Object[][] getContents() {
        return new Object[][]{
            {"hello", "Saluton!"},
            {"bye", new String[]{
                "Ĝis!",
                "Ĝis revido!",
                "Estis agrable vidi vin!"
            }},
            {"animal.name", (UnaryOperator<String>) name -> name},
            {"animal.question", (UnaryOperator<String>) animal ->
                "Ĉu ĝi estas " + animal + "?"}
        };
    }
}
```

As you can see, we can store grammar rules for a particular language in the resource class. Here are all the rules that you need to add to the resource file to support Esperanto:

- Statements must begin with the word **Ĝi**....  
**It is** a mammal = **Ĝi** estas mamulo  
**It can** fly = **Ĝi** povas flugi  
**It has** horns = **Ĝi** havas kornojn
- Negative facts are formed by simply adding **ne** after the word **Ĝi**:  
**It isn't** a mammal = **Ĝi ne** estas mamulo  
**It can't** fly = **Ĝi ne** povas flugi  
**It doesn't have** horns = **Ĝi ne** havas kornojn
- Questions are formed by simply adding the particle **Ĉu** to the beginning of the statement:  
**Is it** a mammal? = **Ĉu** ĝi estas mamulo?  
**Can it** fly? = **Ĉu** ĝi povas flugi?  
**Does it have** horns? = **Ĉu** ĝi havas kornojn?
- There is no indefinite article in Esperanto.  
**a** cat = kato  
**an** ape = simio
- Esperanto, like English, has a definite article.  
**The** cat = **La** kato  
**The** dog = **La** hundo.
- To ask about an animal, we need to write "**Ĉu ĝi estas...**?"  
**Is it** a cat? = **Ĉu ĝi estas** kato?

When you are done with internationalization, you can select the language when you start the program using the "user.language" key of the Java virtual machine. For example, to specify the Esperanto language, you need to run the following command:

```
java -Duser.language=eo Main
```

In the IntelliJ development environment, you can specify this key in the "Run / Debug Configurations" in the "VM Options" field.

## Objective


You need to add support for Esperanto. You can get the translation of phrases in the attached [resource files](#), or you can use [Google Translate](#) to translate from English into Esperanto. You need to implement the correct rules for constructing questions and negative statements. When saving the knowledge tree to the disk, you must add the language prefix to the file name, for example: `animals_eo.json`. In the test directory, you can see the test script in the `esperanto.script.yaml` file.

**HINT** by KS **Knute Snortum** [Viewed hints](#)

If you are using the supplied resource files and you see something like this:

some.message = This is a message with {0} in it



...then look up MessageFormat.

 This is the last hint available for this problem! Please post your own hint after completing the problem, future learners will thank you.

 Write a program

[Code Editor](#)     [IDE](#)

CONNECTION STATUS

-  IDE is responding   IntelliJ IDEA 2022.1.1
-  EduTools plugin is responding   2022.5-2022.1-343

Solve in IDE

 Synchronizing IDE may take a while

[Comments \(31\)](#)

[Hints \(2\)](#)

[Useful links \(0\)](#)

[Solutions \(12\)](#)

[Show discussion](#)