Object-Oriented Programming

Programming Project

Academic Year 2022-2023

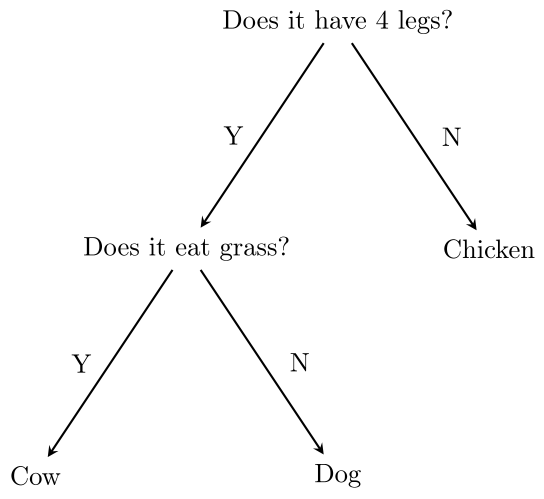
This project can be completed by groups of up to two students and must be submitted to `bvergain@uliege.be` *for April 16th at the latest. Projects returned after the deadline will not be graded.*

This project consists in programming in Java a guessing game, based on a binary decision tree that is incrementally constructed by interacting with the player.

A *binary decision tree* is a data structure that classifies values. It is composed of a finite number of *nodes* that are either *internal* or *leaf* nodes. Each leaf node is labeled with one value. Each internal node is labeled with a yes/no question, and has two *children* nodes: one for the yes and one for the no answers. Every node is the children of exactly one other node, except for the *root* node of the tree which is not the children of any node. The tree cannot contain cycles: a path followed by repeatedly moving from an internal node to one of its children cannot visit a node multiple times.

In order to use such a data structure for classifying a value v, one starts from the root node n and, if this node is internal, checks whether the value v answers positively or negatively the question that labels n. The procedure then continues by replacing n by its appropriate children, and so on until one reaches a leaf node. If v is equal to the value v' that labels this leaf node, then the operation terminates. Otherwise, one creates a new leaf node for the value v, and replaces the leaf node that has been reached in the tree by an internal node labeled with a question that distinguishes v from v'.

For instance, the following binary decision tree classifies animals:



A guessing game based on this tree could have the following interactions with a player. **Note:** Data input by the player is prefixed by “>”.

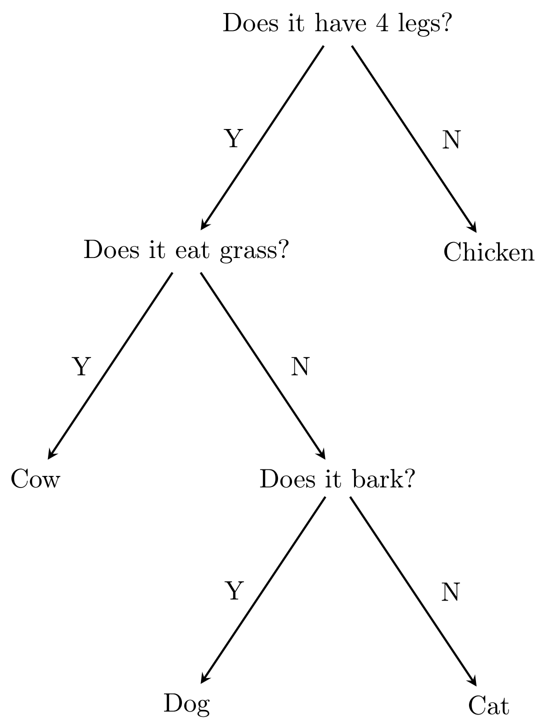
Game 1:

```
Welcome to the game!
Please choose an animal, and then press <return>.
>

Does it have 4 legs (Y/N)?
> Y

Does it eat grass (Y/N)?
> N

Is it a dog (Y/N)?
> Y

I have won!
```

Game 2:

```
Welcome to the game!
Please choose an animal, and then press <return>.
>

Does it have 4 legs (Y/N)?
> Y

Does it eat grass (Y/N)?
> N

Is it a dog (Y/N)?
> N

I am unable to guess; you have won!
What did you choose?
> a cat

What question could I ask to distinguish a cat from a dog?
> Does it bark?

For a cat, would you answer yes or no to this question (Y/N)?
> N

Thank you!
```

After this game, the updated decision tree now looks as follows:



This updated tree could then lead to the following game:

Game 3:

```
Welcome to the game!
Please choose an animal, and then press <return>.
>

Does it have 4 legs (Y/N)?
> Y

Does it eat grass (Y/N)?
> N

Does it bark (Y/N)?
> N

Is it a cat (Y/N)?
> Y

I have won!
```

Your Java program must be able to interact with the player in the same way as those three sample games. This program must store the binary decision tree that it constructs into a file, in order to be able to grow it incrementally over several playing sessions. The format of this file is imposed and must obey the following specification:

- The file is a text file, the first line of which contains the number of nodes of the decision tree, followed by a space and the message to be displayed at the beginning of the game (without the suffix "`, and then press <return>.`").
- Each subsequent line corresponds to one node. The nodes are numbered sequentially from 1, i.e., for every $i \geq 1$, the node with the number i is described by the $(i+1)$-th line of the file.
- The root node (if the tree is not empty) has the number 1.
- A leaf node is represented by a line of the form

  `= Value associated to this node`
- An internal node is represented by a line of the form

  `? n1 n0 Question associated to this node`

  where `n1` and `n0` are the numbers of the children nodes associated (respectively) to the yes and no answers to the question. The question is written without its final question mark.

For instance, a possible content for a file representing the decision tree obtained at the end of Game 2 is the following:

```
7 Please choose an animal
? 2 6 Does it have 4 legs
? 7 4 Does it eat grass
= a dog
? 3 5 Does it bark
= a cat
= a chicken
= a cow
```

Important guidelines

- Your program must only interact with the player in text mode; you are not supposed to implement a graphical user interface.
- It must be possible to start your program by the command

  `java Game filename`

  where `filename` is the name of the file containing the representation of the binary decision tree to be used.

  Before submitting your project, please check that your program works correctly with the sample decision tree described in this document.
- In the case that the file provided to your program is invalid, your program must detect it and exit with an error message indicating clearly the nature of the problem. It is sufficient to check that the format of the file is incorrect; it is not required, for instance, to verify that the represented structure is free from cycles.

- Your submission must take the form of a `.zip` or `.tar.gz` archive containing the Java source code of your project. After extracting the contents of this archive in the current directory, it should be possible to compile your project with the command

  ```
  javac Game.java
  ```

- Your archive should also contain a file `game-tree.txt` containing a representation of an already built decision tree for a game of your choice. You are not limited to classifying animals like in the example given in this document; other interesting possibilities are to guess countries, artists, plants, dishes, . . .
- Your project will be evaluated with Java 8. It is thus important to make sure that your submission does not rely on features introduced in later versions of the Java language.
- The evaluation will take into account not only the correct implementation of the requirements of this problem statement, but also the compliance of your source code to the principes of object-oriented programming, as well as its modularity, readability, simplicity, and portability.
- Except for classes present in the Java Class Library, which you are allowed to use, the entire code of your project must be written by yourselves. All cases of fraud will be sanctioned according to the general policy of the University (cf. `https://www.student.uliege.be/cms/c_11187649/`). This include copying or adapting code obtained from other students, from online sources, or generated by tools such as Copilot.