Ants

made by Jakub Mosakowski,

second year IT student

University of Gdansk

Table of Contents

Description of the Program

Ants is a simple program which simulates anthill and ant behavior. The program is made of eight classes. All of them will be described precisely below.

Libraries used

- Swing graphics library for providing a graphical user interface.

- AWT for handling few events.

Software used

- Nozbe for task management.

- IntelliJ IDE.

- Git version control.

Classes

MainFrame.java

MainFrame is a main class. The task of this class is to manage and create the frame of the application.

Constructor MainFrame creates new Board and JFrame. All JFrame options are set in method setJFrame i.a. size of the window, extended state, close operation.

It is also responsible for setting actions for buttons on the frame:

- Spawn Ant button action – checks if the number of ants on the screen is smaller than the available quantity of them. If yes, creates new ant. Sets the text of the statement in the right corner of the screen, which informs the user if a new ant have occurred.
- Start Again button action – re-sets the values of variables and calls setJFrame, and setTimer methods.
- Options button action – shows options frame (method showOptions).

Last thing which constructor does is setting the timer. The timer starts a new turn every certain time which user set and also updates timer on the toolbox.

NextTurn method moves ants, and after that checks if every leaf is taken by an ant. If yes, this method shows ending frame. If not, objects from anthill are sent to board where GUI is updating and showing again.

Main method fills Max class with values, creates a new instance of Anthill class and starts the frame.

FillMaxClass is getting screen size and passes it along with others variables to Max.

Ending method stops the timer and creates the frame with the summary.

ShowOption shows a new frame with options where user can put custom values. Input is limited to integers. Every variable has a minimum and maximum value. After accepting, values of variables are saved and used in the next game.

ObjectSquare.java

Parent class for every object which could be shown on board. Only methods in this class are getters and setters. It holds important variables, such as position of an object, degree of where an object is directed, or an icon which should be displayed on board and visibility.

Leaf.java

Child class of ObjectSquare. Only differences are icon and name.

Ant.java

Child class of ObjectSquare. The constructor of this class sets a position of an object, name, and icon.

Move method moves ant at random, adjacent field. Ant knows if she can go to certain field, because we are passing all object to this method. If an ant doesn't have a leaf, moves randomly. There is 50% probability that an ant will go straight instead of turning. Before every move, ant checks a field on which she might move if it is not occupied by other ant/queen or if it's not outside of the board. If none of these contraindications are present, an ant moves to this field. If an ant holds a leaf, she moves to another field with a leaf. If an ant doesn't hold a leaf and enters on the field with a leaf, she grabs it. After that ant changes her icon to an ant with a leaf. Leaf is disabled. If an ant has a leaf, she's going straight to the queen instead of random walking. From now, an ant is taking the shortest path to the queen (check what else should be done).

Queen.java

Queen is a child class of an Ant. Every queen has her own unique name.

Max.java

Max is storage class, which holds values like a maximum number of fields, size of a screen, max values of variables etc.

Anthill.java

Anthill contains every object on board and passes it further.

Constructor spawns queen and leaves.

SpawnLeaves randomize values for every new leaf and makes sure every leaf has a distinct position.

SpawnAnt creates new ant next to the queen. The method is performed after pressing the button on the toolbar (Spawn Ant).

MoveAnts moves every ant. Furthermore, it checks if a leaf was grabbed, if yes then anthill disables that leaf.

Anthill has a counter for leaves, which is used for displaying a number of leaves on the toolbar. In every turn MainFrame checks if all leaves are collected.

Board.java

It is a class for drawing whole board and toolbox.

The constructor initializes GUI i.e. setts the toolbar, creates squares and fills the board.

SetToolbar adds toolbar to the GUI. Toolbar contains three buttons and four labels.

CreateSquares method sets empty squares with empty image and assigns them to a certain position on board.

FillBoard removes everything from board and adds updated images to correct positions.

SquareWithImage first checks if the passed object is visible and checks if it is not an empty object. If it is then it passes an empty image to a certain position. But if none of the conditions is met, then it gets an image from an object and passes it to the rotateImage method. After that sets the image to a certain position.

RotateImage gets degrees where an object is directed and then turns the image in the right direction.

ShowNewTurn first sets every panel to empty one. Then gets every object (which was earlier passed) and sets it on its correct position (if an object is set to visible).

What else should be done:

- Move some methods to a new classes. Showing options could be a new class.

- Queen shouldn’t be a child class of an Ant.

- Ant leaves leaf close to the queen and is going to search for a next leaf.

- A leaf which was abandoned should not be able to be grabbed again.

- Ant should find a different path if the shortest is blocked.