Greenfoot hints 📗👣

Basic

-How to use the Scenario Editor?

1. You create an object by right-clicking an *Actor-subclass* in the right hand object tree.
2. Drop the image by left-clicking it in the game area.
3. Right-click the newly dropped object to access its properties.
4. Remember: When you want to save it as code, choose “Save the World”!

Tab/shift+Tab - quickly indent paragraph

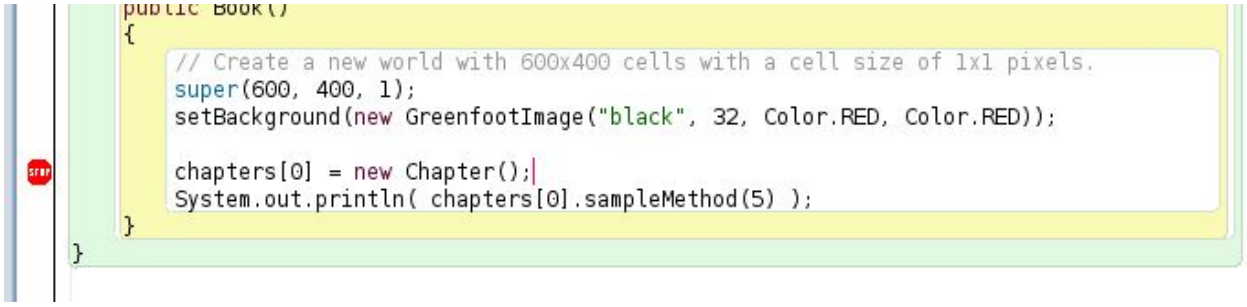
You can select a number of lines of code and toggle indentation to keep code tidy!

F8/F7 - quickly comment out code (hide/disable)

You can select a number of lines of code and toggle between `//act();` and `act();` !

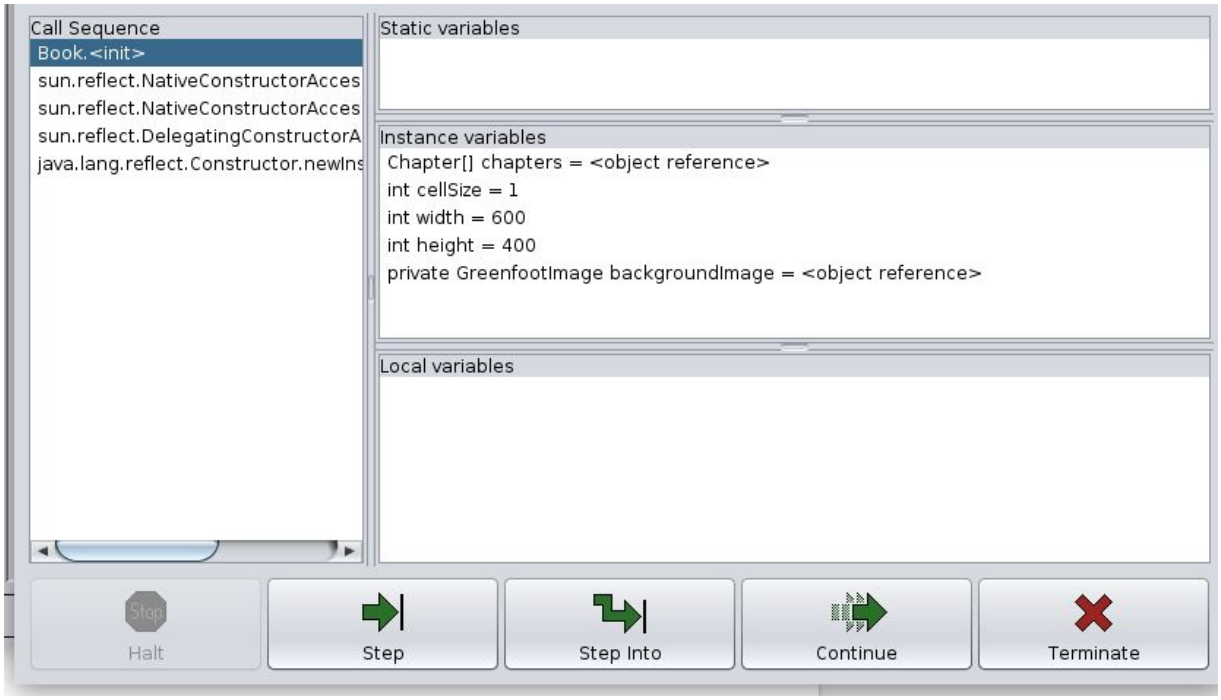
Breakpoints - pauses the game on a specific line!

- Learn how your objects and variables change and behave in runtime
- Find bugs quicker



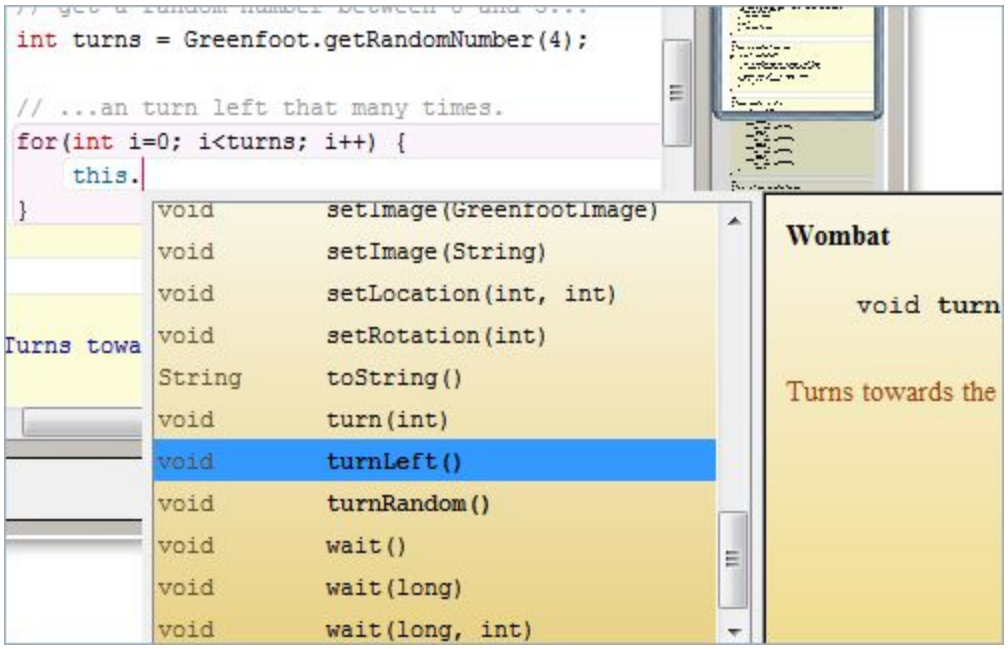
How to use:

- Clicking in the left margin of a compiling file.
- Set the cursor on preferred line and press `Ctrl+B` / `cmd+B`
- The breakpoint will be active immediately, but you might have to reset the game. The game pauses and the “Debugging window” appears:



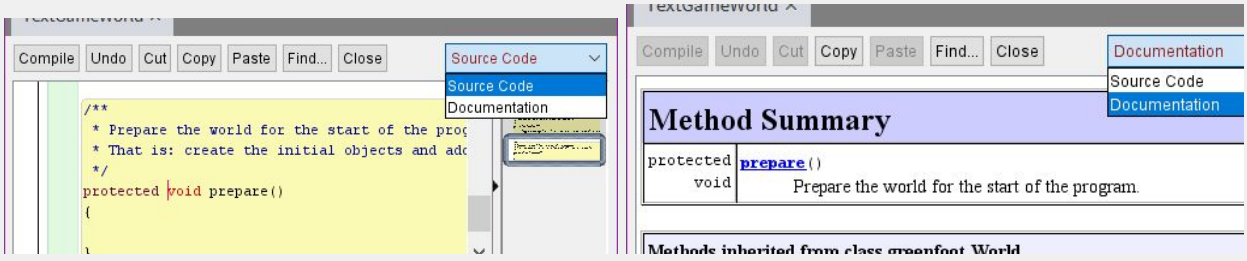
- To observe next time same breakpoint is reached, use “Continue”
- More than one breakpoint can be added
- “Terminate” will cause Greenfoot to crash/restart and is not recommended.
- If you experience trouble caused by debugger, it should help to close and reopen Greenfoot manually.

CTRL+Space - quick code lookup and completion



- When CTRL+space is hit while cursor is set next to an instance with dot, like the example image above, the list of inherited methods will be shown (cf [tutorial part 5s5])
- When cursor is on an empty line suggestions will show static methods from Greenfoot API documentation. For example `setBackground()` , `getObjects()`
- To access other java methods and classes, you need to add import statement for that package and compile first! For example `import java.lang.Arrays;` , `import java.time.*;`

Comment blocks (/* ... */) turn into documentation



- Can be used on classes, methods and member-variables (the ones that can be made “public”, not local ones of the inner code blocks)
- Anything “public” or “protected” will appear in docs automatically
- Regardless of “private” or “public” it will still appear when using CTRL+space at proper position and you start typing the name

Intermediate

-How to delay things?

You use something called timer (or timeout). In Greenfoot Scenario Editor, this is a class that can be imported by right-clicking the area around the object-tree and selecting “SimpleTimer”. Look in the description for further instructions!

-How to move things slower than one pixel per act?

You . In Greenfoot, this is an object that can be imported by right-clicking the area around the object tree in Scenario Editor. Choose “SmoothMover”. Then create subclass from it instead of Actor. They will work the same as Actor, except that “move”, “turn” and other methods can accept numbers using decimals. So, using “move(0.5)” will result in a slower movement.

Troubleshooting

- I opened a zipped scenario and got this:

“Exception in thread "JavaFX Application Thread" java.lang.NullPointerException”

- This probably means you already opened it. To be able to open it again, remove the folder created by Greenfoot, with the name of the scenario. Or, open it from another folder.

- The Scenario Editor is not updating my code anymore, why?

You are only able to save levels created in editor as long as you don't make any "intrusive" changes in your code. That is, inside `public SubClassedWorld(){ }` and inside `prepare(){}`. You can however do some cleaning inside `prepare(){}` by removing unnecessary things, or changing parameters in code. Those are non-intrusive, and it's a good practice to figure out how to do this even if it's fine to do "intrusive" changes for worlds where you can't use or don't need the Scenario Editor.

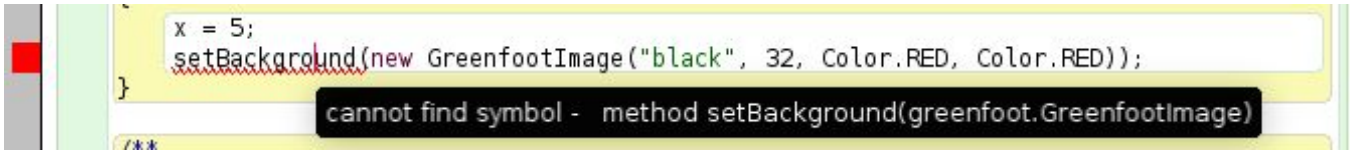
- Why are the coordinates referring to center of image at times, and other times to upper left corner?

There are usually more than one coordinate system in game development, where different rules apply:

1. World coordinates (aka Model-coordinates) using metric system and higher precision units.
    a. setLocation (Actor)
    b. getX, getY (Actor)
    c. addObject (World)
2. Screen coordinates (aka Window-coordinates) using pixels and integers
    a. drawImage, drawLine, drawString, etc. (GreenfootImage)
    b. MouseInfo.getX, MouseInfo.getY (Greenfoot)

In greenfoot, the default settings simplify things by using whole pixels and integers for both above systems. Still, in order to simplify positioning things in the world and drawing things on the screen respectively, it is generally preferred to treat them slightly differently.

- **I use same code in my own class as example, but get this:**



"Cannot find symbol" commonly indicates a missing import-statement. In this case, however, it's not possible to use "setBackground" in a class that is not a subclass of "World".

- Greenfoot refuses to do what it should, or did I do something wrong?

Greenfoot is not perfect, but when it refuses to do something it normally should or if it behaves buggy overall, it usually helps to restart the whole program.