Programming in Java

Exam project information

Prof. Dr. Lesley De Cruz

Version of 20 October 2025

1 **Introduction**

2 **Evaluation criteria**

3 **Documentation: Javadoc**

Exam project for Programming in Java

Your challenge:

Write a text adventure game in Java, that meets the specified criteria.

Group project

For this project you will work in groups of two.

- Random assignment
- Group work will be done using git (covered in WPO)
- Good communication is important! If problems come up, try to talk about them as fast as possible.
- If serious problems persist in the group, even after good communication, contact us quickly so we know of the problem(s) and can take this into account.

Deadlines and oral exam

Attention!

Intermediate submissions are **mandatory** but **not graded**.

1. **November 9th** (before you start writing your code): submit a **UML class diagram** of your code architecture and a short description.
2. **December 7nd**: submit a first version **v1** of your project.
3. **January 4th**: final submission (graded) **v2** of your project.
4. **January [TBD]**: oral exam: **Q&A** on campus
   We will assess your understanding of Programming in Java by asking questions related to your project.
   You can book your time slot in the calendar section of Canvas.

How to submit your project

Submit your code by uploading a **project** via CANVAS. Please mind your project name!

- The project should only contain the '.java' files and the package structure, no other IDE specific folders should be uploaded.
- Please make sure your project is named as follows: **SURNAME_givenName_gameName_version**
- Make sure this is the **name of your project**, not just the name of the folder or zip file!
- To **rename** your project, right-click the project name in the Package Explorer and choose **Refactor -> Rename...**
- The metadata and directory name will be adapted automatically

How to rename a project

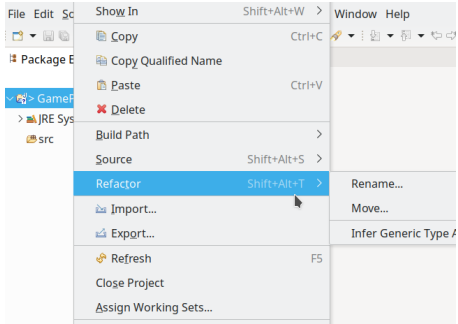


Figure: Eclipse

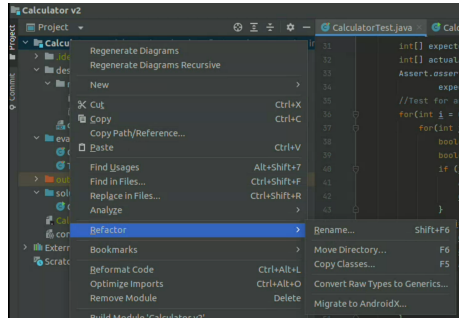


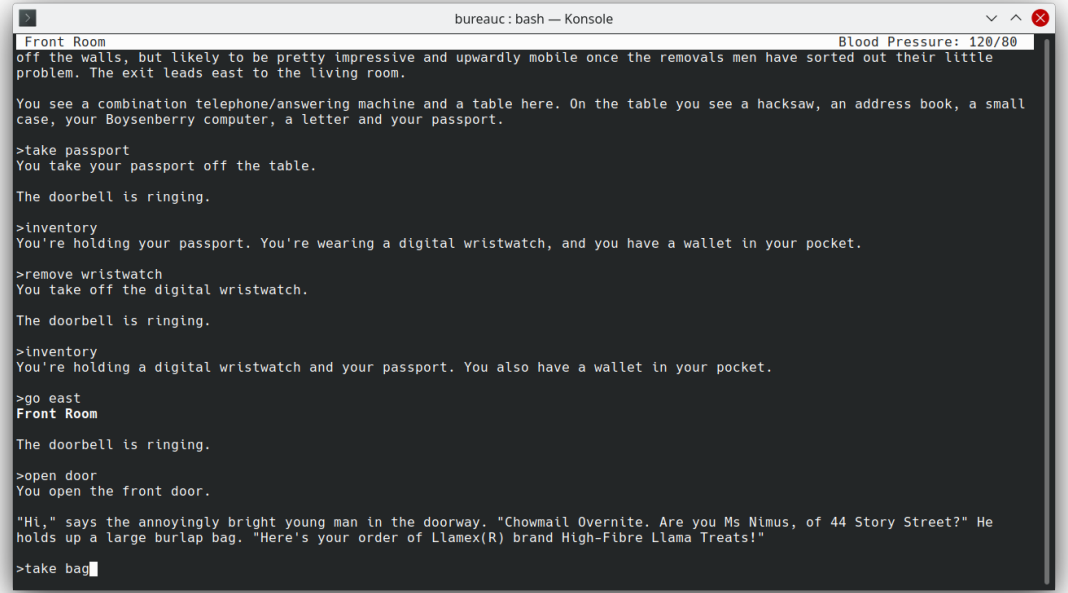
Figure: IntelliJ

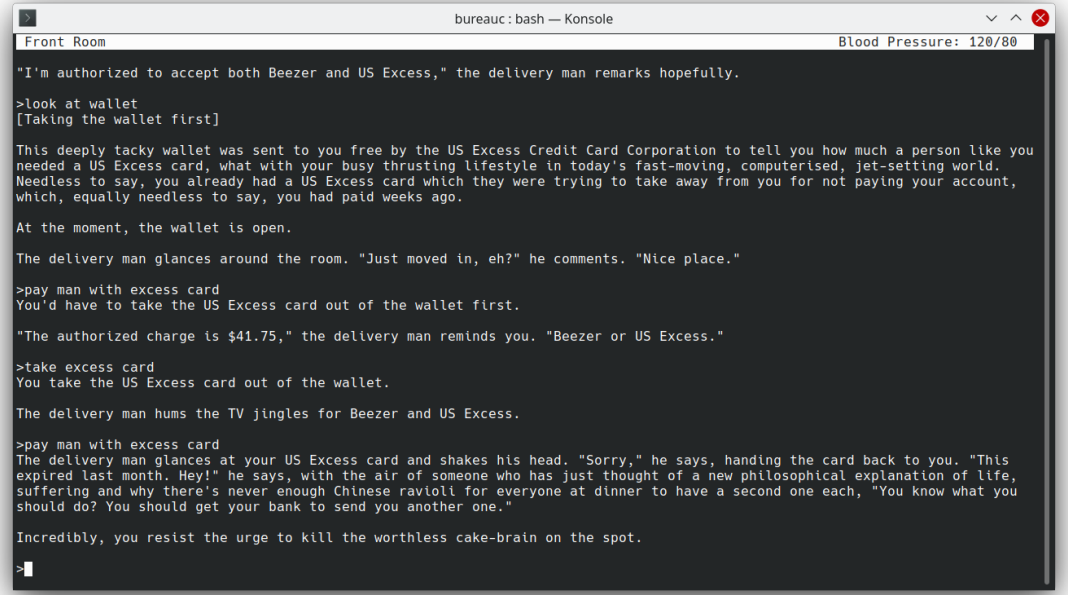
Example 1: Bureaucracy

Play online: `https://archive.org/details/msdos_Bureaucracy_1987`

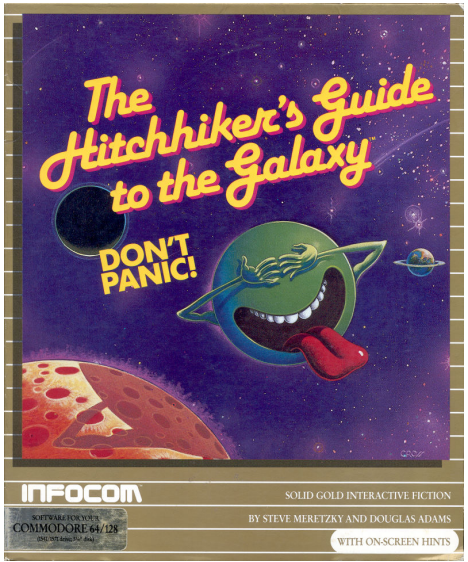
Bureaucracy is a text adventure game that describes the misadventures of a person who has recently moved to a new apartment and begins to fall victim to various bureaucratic procedures that always go wrong. He can't receive his mail, access his bank account, or generally lead a normal life; all his activities are impeded by needless and confusing hurdles. In order to retrieve the lost mail and put his life back on track, the protagonist will have to meet exotic characters and perform various tasks while being constantly threatened by various bureaucratic occurrences.

The gameplay involves typing commands composed out of verb and object combinations, used to interact with the game world and solve puzzles. In addition, a special "blood pressure" meter will gradually fill itself the more the protagonist is bothered by bureaucratic annoyances. Once this meter is completely full, the player character suffers an aneurysm, and the game ends.


bureauc : bash — Konsole
Front Room
Blood Pressure: 120/80
off the walls, but likely to be pretty impressive and upwardly mobile once the removals men have sorted out their little problem. The exit leads east to the living room.
You see a combination telephone/answering machine and a table here. On the table you see a hacksaw, an address book, a small case, your Boysenberry computer, a letter and your passport.
>take passport
You take your passport off the table.
The doorbell is ringing.
>inventory
You're holding your passport. You're wearing a digital wristwatch, and you have a wallet in your pocket.
>remove wristwatch
You take off the digital wristwatch.
The doorbell is ringing.
>inventory
You're holding a digital wristwatch and your passport. You also have a wallet in your pocket.
>go east
Front Room
The doorbell is ringing.
>open door
You open the front door.
"Hi," says the annoyingly bright young man in the doorway. "Chowmail Overnite. Are you Ms Nimus, of 44 Story Street?" He holds up a large burlap bag. "Here's your order of Llamex(R) brand High-Fibre Llama Treats!"
>take bag


bureauc : bash — Konsole
Front Room
Blood Pressure: 120/80
"I'm authorized to accept both Beezer and US Excess," the delivery man remarks hopefully.
>look at wallet
[Taking the wallet first]
This deeply tacky wallet was sent to you free by the US Excess Credit Card Corporation to tell you how much a person like you needed a US Excess card, what with your busy thrusting lifestyle in today's fast-moving, computerised, jet-setting world. Needless to say, you already had a US Excess card which they were trying to take away from you for not paying your account, which, equally needless to say, you had paid weeks ago.
At the moment, the wallet is open.
The delivery man glances around the room. "Just moved in, eh?" he comments. "Nice place."
>pay man with excess card
You'd have to take the US Excess card out of the wallet first.
"The authorized charge is $41.75," the delivery man reminds you. "Beezer or US Excess."
>take excess card
You take the US Excess card out of the wallet.
The delivery man hums the TV jingles for Beezer and US Excess.
>pay man with excess card
The delivery man glances at your US Excess card and shakes his head. "Sorry," he says, handing the card back to you. "This expired last month. Hey!" he says, with the air of someone who has just thought of a new philosophical explanation of life, suffering and why there's never enough Chinese ravioli for everyone at dinner to have a second one each, "You know what you should do? You should get your bank to send you another one."
Incredibly, you resist the urge to kill the worthless cake-brain on the spot.
>

Example 2: The Hitchhiker's Guide to the Galaxy

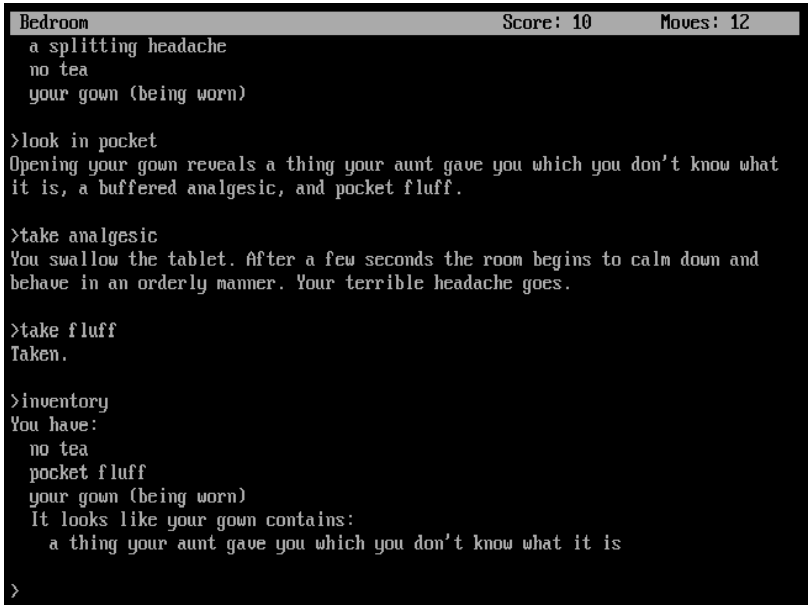


Example 2: The Hitchhiker's Guide to the Galaxy

Play online: `https://archive.org/details/msdos_Hitchhikers_Guide_to_the_Galaxy_The_1984`

You are Arthur Dent, an Englishman with a bad hangover wearing a dressing gown containing a much needed buffered analgesic and some fluff. Your house has just been destroyed, followed shortly thereafter by your planet Earth (mostly harmless). You've been rescued by your friend Ford Prefect, who's not actually an out-of-work actor. He has given you a book (The Hitchhiker's Guide to the Galaxy), a towel, and is now telling you to put a fish in your ear. It must be a Thursday; you've never quite gotten the hang of Thursdays.

The Hitchhiker's Guide to the Galaxy is written by Douglas Adams and Steven Meretzky and based on Adams' BBC radio series, television series, and the series of subsequent novelizations. It's one of the classic Interactive Fiction games produced by Infocom, labeled as Science Fiction and has a Standard Level of Difficulty. Though divergent from the source material, the main characters, locations, and concepts are here. Unlike the book, death can come quickly if Arthur fails to observe his surroundings, collect inventory, talk to people, and consult the Guide. DON'T PANIC!


Bedroom Score: 10 Moves: 12
a splitting headache
no tea
your gown (being worn)
>look in pocket
Opening your gown reveals a thing your aunt gave you which you don't know what it is, a buffered analgesic, and pocket fluff.
>take analgesic
You swallow the tablet. After a few seconds the room begins to calm down and behave in an orderly manner. Your terrible headache goes.
>take fluff
Taken.
>inventory
You have:
no tea
pocket fluff
your gown (being worn)
It looks like your gown contains:
a thing your aunt gave you which you don't know what it is
>

Example 3: T-Zero

Play online: `https://archive.org/details/msdos_T-Zero_1991`

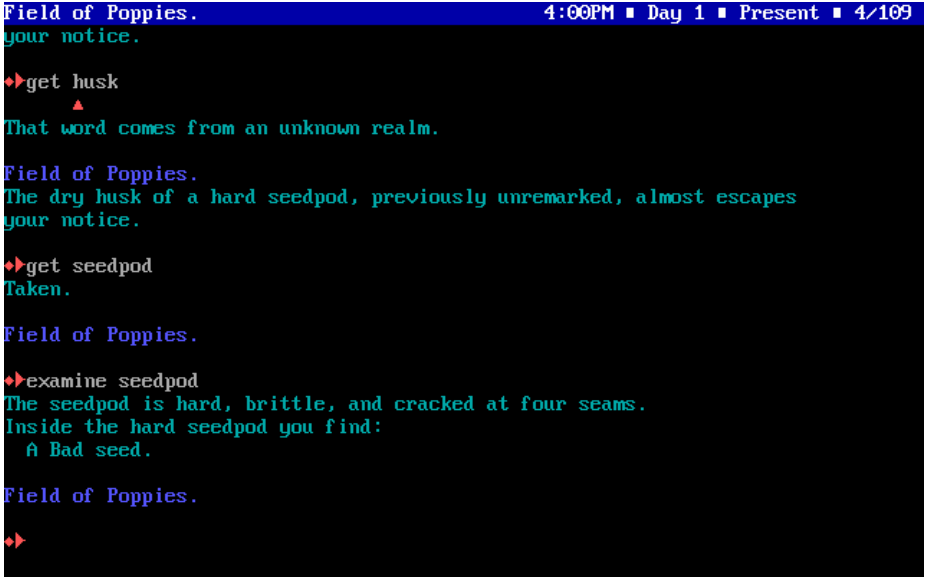
T-Zero is a highly unusual piece of epic interactive fiction, playing in a Lewis Carroll-like surreal surrounding, with a strong emphasis on word puzzles.

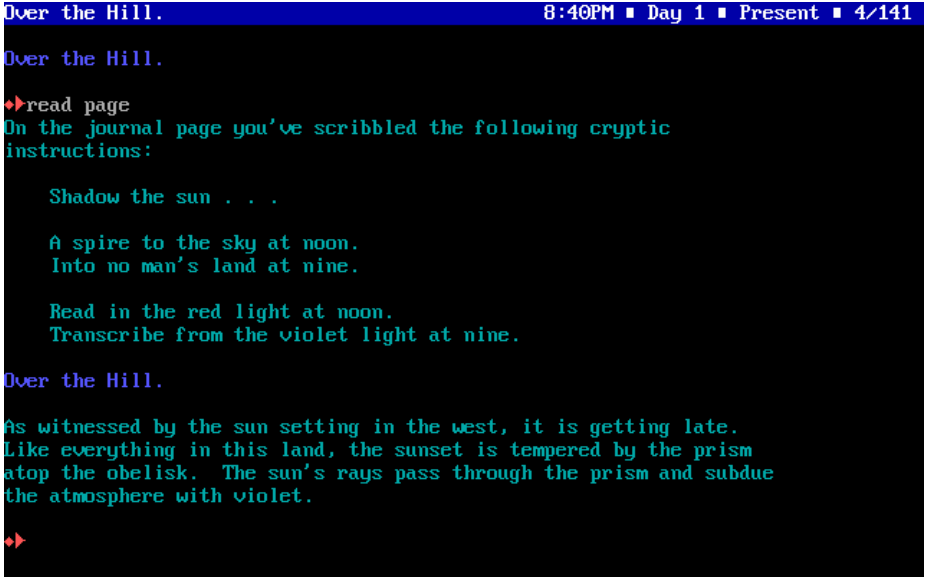
The game starts with you awakening in a River Bed. You have to explore your surroundings to find six objects, "scattered across ages and landscapes", to transport them "to progressively future time zones where they can right the troubled times."

The puzzles are quite hard, as they often use puns (as in Infocom's Nord and Bert) or elements of Western pop culture (like the Beatles). Non-native English speakers could have a hard time to solve this game.

The parser of T-Zero is quite unusual, powerful but tricky. It understands uncommon verbs like IMAGINE (this one allows you to visualize objects and locations have not encountered). Still, it sometimes is quite touchy and requests you to type the exact phrase it wants.

The game also contains a hint system.


Field of Poppies. 4:00PM ■ Day 1 ■ Present ■ 4/109
your notice.
get husk
That word comes from an unknown realm.
Field of Poppies.
The dry husk of a hard seedpod, previously unremarked, almost escapes your notice.
get seedpod
Taken.
Field of Poppies.
examine seedpod
The seedpod is hard, brittle, and cracked at four seams.
Inside the hard seedpod you find:
A Bad seed.
Field of Poppies.


Over the Hill. 8:40PM ▪ Day 1 ▪ Present ▪ 4/141
Over the Hill.
read page
On the journal page you've scribbled the following cryptic instructions:
Shadow the sun . . .
A spire to the sky at noon.
Into no man's land at nine.
Read in the red light at noon.
Transcribe from the violet light at nine.
Over the Hill.
As witnessed by the sun setting in the west, it is getting late. Like everything in this land, the sunset is tempered by the prism atop the obelisk. The sun's rays pass through the prism and subdue the atmosphere with violet.

Rule number 1: no plagiarism!

Clearly state what part of the project is your own work, AND what is not!

- Your grade will only be determined by the part of your project that is your **own, original work**.
- If you do use classes, libraries or code fragments from other sources, make sure this is **indicated very clearly** in the comments.
  This includes generative AI such as chatGPT!
- Submitting code that is not your own work without mentioning this, is considered **plagiarism**.
- **Plagiarism is taken very seriously**, and may lead to disciplinary action including termination of enrolment and exclusion from the institution.
- For more information, see the VUB examination regulations:
  `https://www.vub.be/sites/default/files/2025-07/2025_Reglementen_OER_2025-2026_ENG.pdf`

Evaluation criteria

1. Code readability
2. Code design
3. Good use of object-oriented programming
4. Documentation
5. Functionality and completeness
6. Creativity
7. Oral exam

PRO TIP:

Read the detailed evaluation criteria (1-6) and their weights in the **rubric** of the corresponding Canvas assignment!

Code readability

- Use understandable and clear **names** for variables, methods and classes.
- Follow the conventions of **CamelCase**, with the first letter:
  uppercase for a `ClassName` and
  lowercase for a `variableName` and `methodName`.
- Avoid confusing or ambiguous code, e.g. *shadowing* of instance variables
- **Structure** your code in a logical way.
- Proper use of **indentation** and line length.
- **Comment** your code, especially non-trivial parts.
- Create multiple smaller logical methods, **not** one huge **monolithic** method

Code design

- Avoid **redundancy**: duplicate code means you're doing something wrong. Redundancy can be avoided by using e.g.
  - loops
  - methods
  - inheritance
  - interfaces
- Avoid "magic numbers" that are hardcoded (multiple times) in your code. Instead use **constants** (`static final` fields).
- Use ***enums*** to give a meaning to a set of values.
- Minimize the use of the **`static`** keyword (even if the compiler suggests this, it doesn't mean it's good design).
- Proper use of **access** modifiers (`private`, `public`, ...)
- Write **robust** code by handling Exceptions where/if appropriate.

Good use of object-oriented programming

- **Logical organization** of the code in various classes
- Each class has **instance variables/fields** that characterize this class.
- Each class has suitable **methods**.
- A suitable **class hierarchy**. If required: with one or more abstract classes
- Proper use of **encapsulation**.
- Minimize the **scope** of variables.
- The code should be **easily extensible**: adding a new type of character or item should not require adapting the code everywhere
- Don’t create too many **dependencies** between classes.
- Ask yourself: Is a new class needed or can I work with a new instance of a class?

Documentation

- Provide documentation for **classes**, **methods** and instance variables.
- Use the **Javadoc** syntax for comments:
  - Multiline comments start with `/**`
  - Specific meta-data are described by **inline** and **block tags** (marked with `@`)
  - Specify method parameters with `@param` and return values with `@return`.

Functionality and completeness

- The program can be run and works as required
- The player can play the game by entering commands (e.g. “take sword”)
- The game runs robustly even if wrong input is given (error handling)
- All requested features are present in the program (see next slides)

Functionality and completeness: requirements

Your program should have at least:

- A class representing a **location**:
  The player should be able to **look** at a location, and **move** from one location to another (e.g. “go south”)
- A class representing a **character**:
  The player should be able to interact with a character, you are free to choose the type of interactions (**fight**, **heal**, ...). Dialogs are not required (too difficult).
- A class representing an **item**:
  The player should be able to **take** and **drop** items, **use** them, and have multiple items in an **inventory**.
  You can have different kinds of items such as weapons, potions, food...

Functionality and completeness: how we will test it

We will test your code by entering the following commands, and to verify if code runs as expected (and does not crash)

- go north, go east, go south, go west
- inventory
- look
- help
- take [item_name]
- asfdsadsadfs
- ...

⇒ Test your code extensively and think about user input and output (see previous WPO)!

Creativity

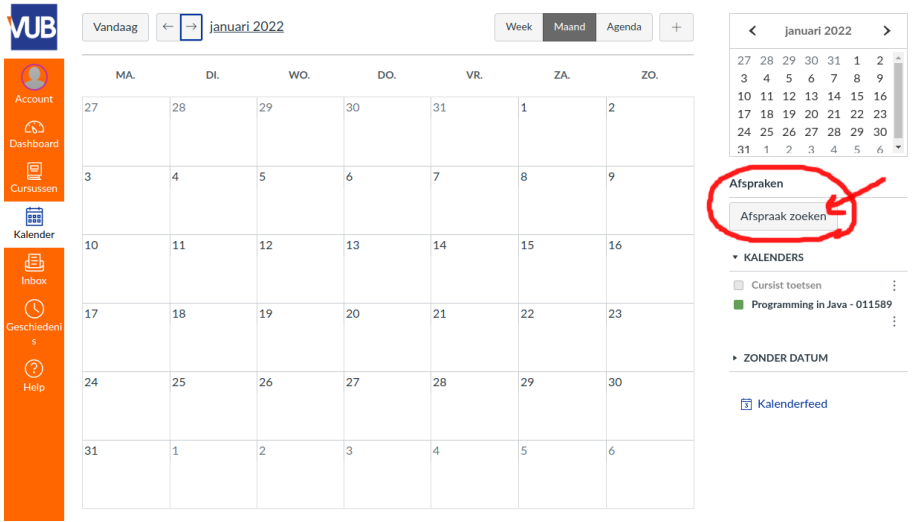
- One tip: **keep it simple!**
- Your focus should be on programming, **not** on the contents of the game.
- *However, a game that meets the technical criteria **and** is particularly original or entertaining will get a bonus point.*

Oral exam

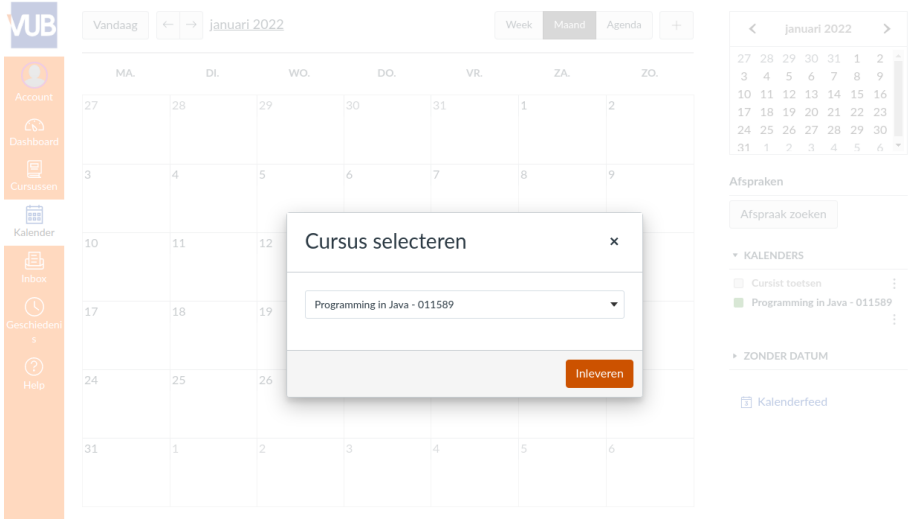
TBD January

- You can book a time slot in the Calendar section of Canvas (see next slides)
- **Do not prepare a presentation**, the oral exam will be Q&A only!
- The oral exam consists of a 20-minute Q&A about your project (design choices, implementation...)
- We will assess your understanding of programming in Java by asking questions related to your project.

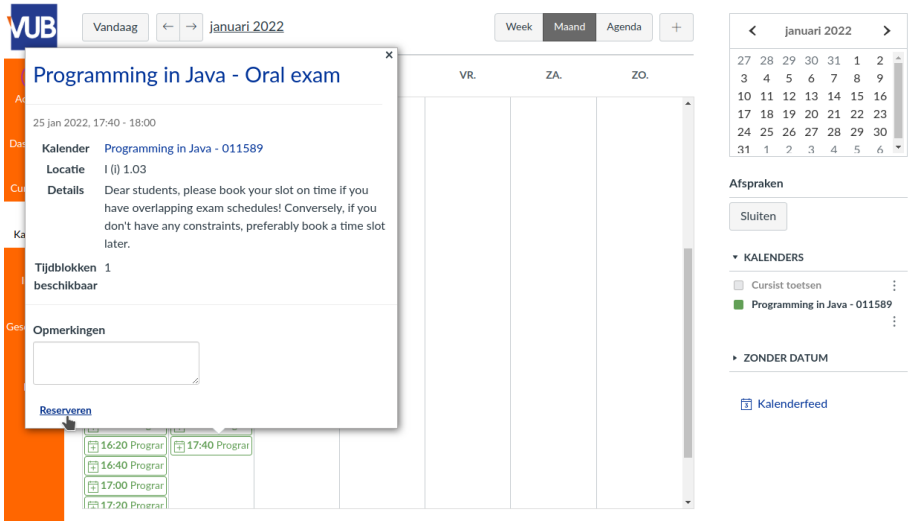
How to book a time slot on Canvas



How to book a time slot



How to book a time slot



Documentation with Javadoc

- Provide documentation for each **class**, **method** and instance variable.
- Use the **Javadoc** syntax for comments:
  - Multiline comments start with **/****
  - Specific meta-data are described by **inline** and **block tags** (marked with **@**)

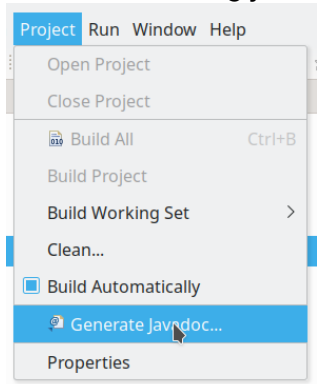
```
/**
 * This method / class / variable does this and that.
 *
 * The first line above is used as a short description,
 * a more detailed description follows underneath. Tags
 * start with an 'at' sign and can be inline:
 * {@link java.awt}. You can also use block tags like:
 *
 * @author Lesley De Cruz
 *
 */
```

Javadoc example

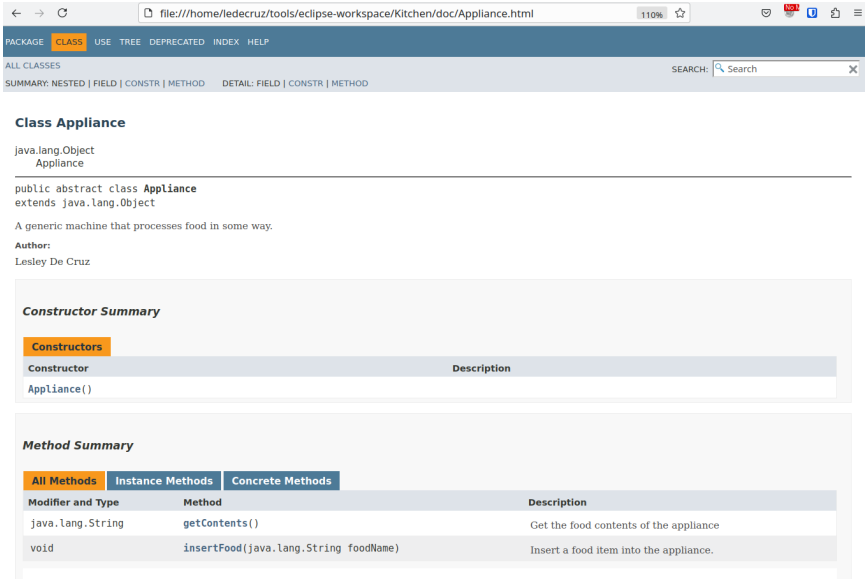
WorldGreeter.java | Toaster.java | Appliance.java

```
1  /**
2   * A generic machine that processes food in some way.
3   *
4   * @author Lesley De Cruz
5   *
6   */
7  public abstract class Appliance
8  {
9      /**
10      * The edible contents of the machine.
11      */
12     private String contents;
13
14     /**
15      * Insert a food item into the appliance.
16      * @param foodName The name of the food to be inserted
17      */
18     public void insertFood(String foodName)
19     {
20         contents = foodName;
21     }
22     /**
23      * Get the food contents of the appliance
24      * @return the food contents of the appliance
25      */
26     public String getContents()
27     {
28         return contents;
29     }
30
31 }
32
```

In Eclipse, build HTML from comments using javadoc:

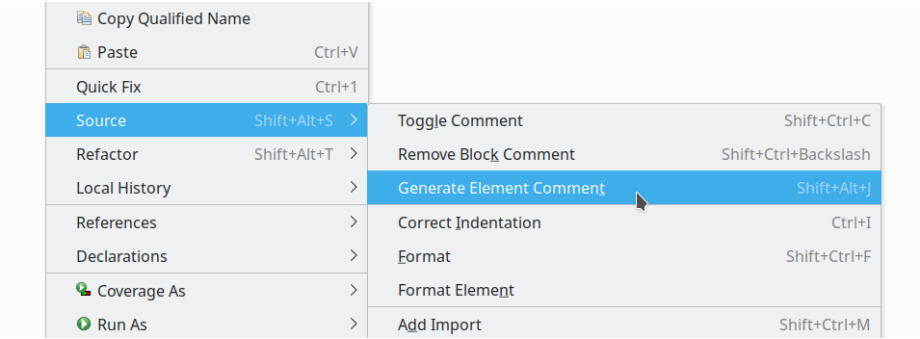


Javadoc generated html



Javadoc in Eclipse

In Eclipse, you can **generate** Javadoc-compatible stubs that you can fill in.
Right-click the class, method or variable name in the editor, and select:
Source > **Generate Element Comment** (or press Shift+Alt+J)



This will automatically insert a Javadoc-style comment above the declaration.

Recap

For the final grade we look at:

- An **individual** report of 1-2 pages. This contains a brief introduction how to play your game together with information about the group working and the choices made to design the game. Some parts of the report can be the same but the **group working** should be written **individually**.
- A working version of a text based adventure game
- Correctly working user input/output
- Fulfilment of the evaluation criteria
- **Bonus points** can be earned by extending the game or is particularly original. Can only be obtained if the rest of the game is working correctly!
- **Individual oral defence** of your project in the exam period

Good luck!