

JavaCraft

Final Report: Group 43

Saturday, October 7, 2023

Table of contents

1	Introduction	2
2	JavaCraft's Workflow	3
3	Functionality Exploration	3
4	Finite State Automata (FSA) Design	5
5	Git Collaboration & Version Control	7
6	Extending the Game Code	7
7	Interacting with Flags API (For Final Submission)	8
8	Conclusion	9
9	References	9
10	Appendix	10

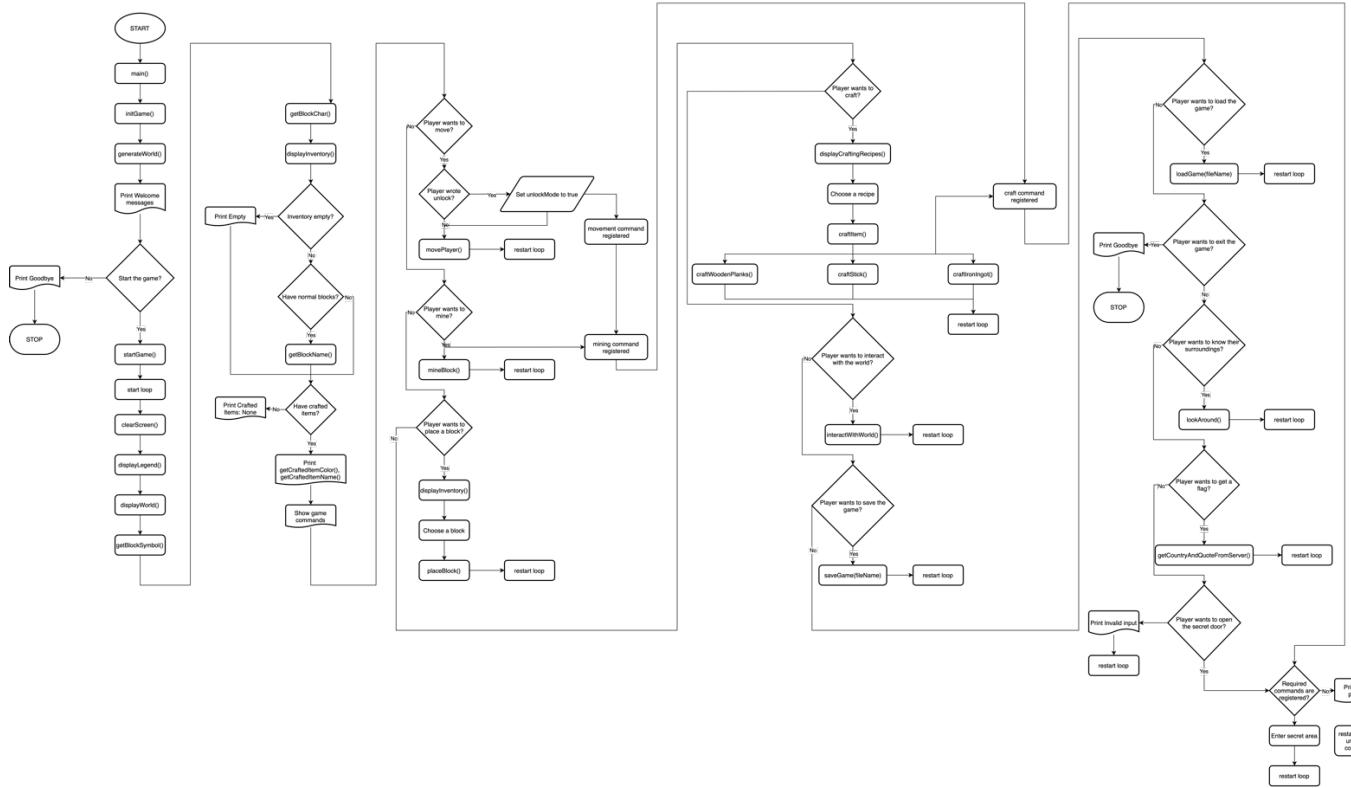
Attribute	Details
Group Name	Syntax Error
Group Number	43
TA	José Manuel Ros
Student Name	Student ID
Botond Moksony	[I6365627]
Vicente Muñoz	[I6346464]
Zhili Yang Wu	[I6362675]
Malo Coquin	[I6302990]

1 Introduction

Name	Tasks
Vicente Muñoz	<ul style="list-style-type: none">- Function definitions (25%)- Flowcharts (20%)- Pseudocodes (20%)- Game flowchart (100%)- FSA illustration & design (33%)- Git student (40%)- Git trouble summary (100%)- Flag (80%)- Added blocks (100%)- Raw meat(100%)- Fix border colors (100%)
Botond Moksony	<ul style="list-style-type: none">- Function definitions (25%)- Flowcharts (30%)- Pseudocode (30%)- Appendix (100%)- FSA illustration & design (33%)- Git teacher (100%)- Report final revision (100%)- Flag (20%)- Cow (50%)
Zhili Yang Wu	<ul style="list-style-type: none">- Function definitions (25%)- Flowcharts (40%) (Did startGame flowchart)- Pseudocodes (50%)- Game pseudocode (30%)- FSA description (100%)- FSA illustration & design (13%)- Git student (40%)- Cooked meat (100%)- Cow (50%)- Eat meat (50%)
Malo Coquin	<ul style="list-style-type: none">- Function definitions (25%)- Flowcharts (10%)- Game pseudocode (70%)- FSA illustration & design (20%)- SourceTree student (20%)- Added crafting recipes (100%)- Eat meat (50%)

2 JavaCraft's Workflow

- Flowchart For Game (also included in the appendix in higher resolution):



- The pseudocode for the game can be found in the appendix.

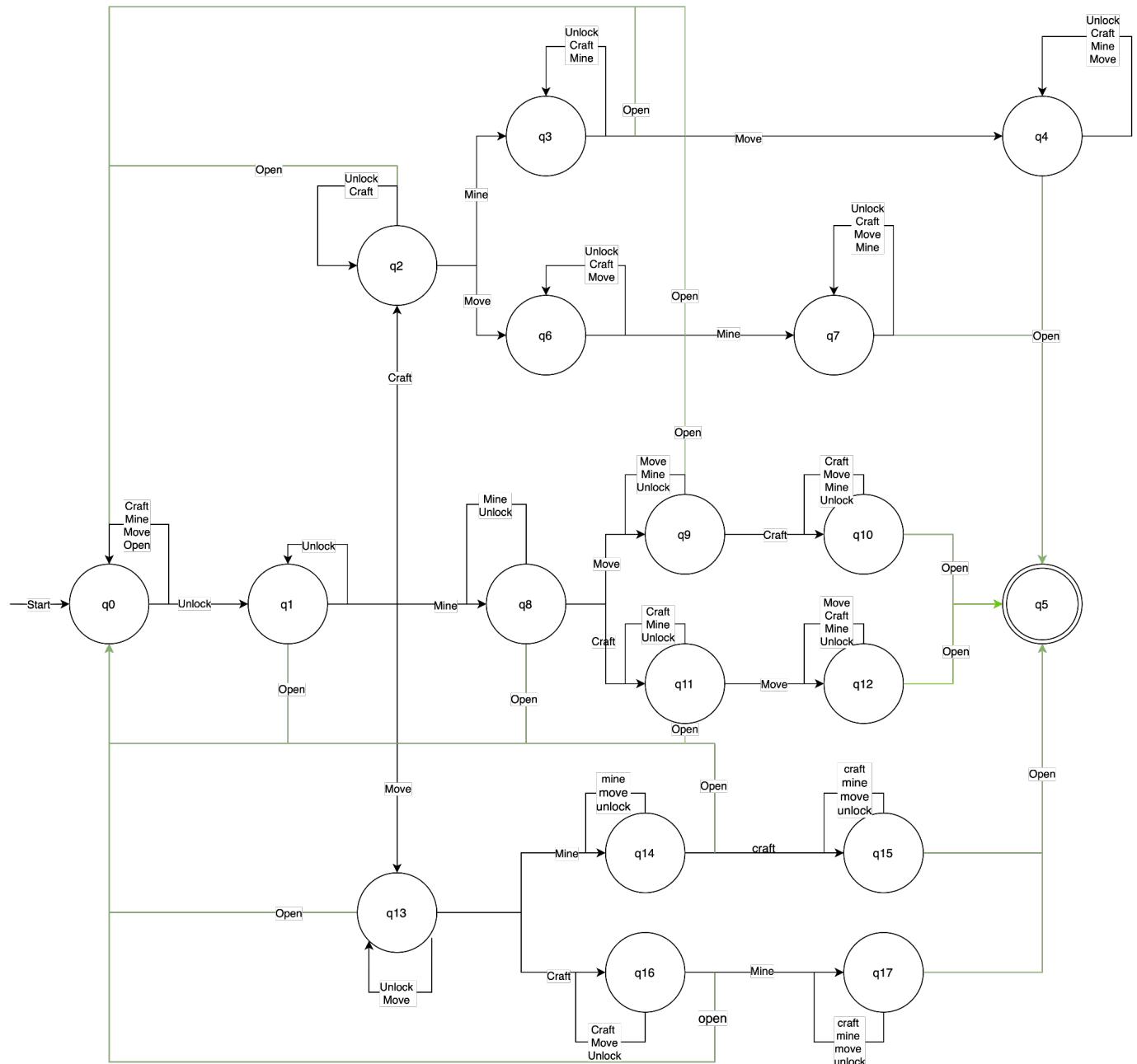
3 Functionality Exploration

List of key functionalities explored:

No .	Function Name	Description
1	main	Print out the game's description and initiate the game after user confirmation
2	initGame	create the world width, set the initial position of the player & create the inventory
3	generateWorld	generate blocks for all the coordinates on the map based on random values
4	displayWorld	display map on terminal
5	getBlockSymbol	function for getting block colors and symbols (some blocks are represented by symbols, e.g.: \$)
6	getBlockChar	function for getting the characters for the blocks
7	startGame	ask for the action, & call a function for the specific action we choose. It also can open the secret door but only with some conditions.
8	fillInventory	fills up the inventory with all the existing blocktypes
	resetWorld	calls for the dutch flag function (generateEmptyWorld) and moves the player to the middle of the screen
9	generateEmptyWorld	Generates the Dutch flag
10	clearScreen	clear the screen according to the os type

11	lookAround	prints out the blocks around the player
12	movePlayer	Changes player coordinates according to the input
13	mineBlock	It allows the player to mine the blocks to get materials if the specified coordinate is not air
14	placeBlock	It allows the player to put a material in a specific coordinate
15	getBlockTypeFromCraftedItem	Return the blockType from the itemID
16	getCraftedItemFromBlockType	Return the itemID from the blockType
17	displayCraftingRecipes	Prints out the crafting recipes.
18	craftItem	It allows the player to use their materials to create an item.
19	craftWoodenPlanks	It allows the player to use the wood they collected in the game and turn it into wooden planks
20	craftStick	It allows the player to use the wood they collected in the game and turn it into a stick
21	craftIronIngot	It allows the player to use the iron ore they collected in the game and turn it into iron ingots
22	inventoryContains	Checks whether the inventory contains a specific item.
23	inventoryContains	Checks whether the inventory contains a certain amount of items
24	removeItemsFromInventory	Remove a specified number of items from the inventory
25	addCraftedItem	Add itemID to the craftedItems list
26	interactWithWorld	Add block to inventory according to the place where the player stands
27	saveGame	Save the game variables, objects to an external file
28	loadGame	Load variables and objects from a previously saved game file
29	getBlockName	Returns the name of the block from blockType
30	displayLegend	print the legend with the symbols and names of the blocks
31	displayInventory	print the player's inventory with the number of blocks that the player has
32	getBlockColor	return the color of each block
33	waitForEnter	It tells the player to "press enter to continue" whenever they need to, and waits for input.
34	getCraftedItemName	return the name of the item from craftedItem
35	getCraftedItemColor	return the color off block, but only for the iron ingot (brown).
36	getCountryAndQuoteFromServer	Communicates with an external server in order to get a flag of a country and a quote besides it.

4 Finite State Automata (FSA) Design



The secret door is a hidden option within the game that is activated when entering the following commands into the terminal:

- Unlock
- Mine (user has to input “m” in either upper or lower case in order to mine)
- Move (user has to either character from WASD in upper or lower case for moving)
- Craft (user has to select any crafting option in this command)
- Open

For opening the secret door function, the user needs to input these commands in specific orders, where each command is entered one and without any repetition in the following inputs. The main conditions needed to enter the sequence are the following:

- The Unlock command always needs to be the first command to be inputted (although you can input this command in intermediate steps as we will mention later).
- Open command has to be strictly the last command to be inputted.
- In between the first input (unlock) and the last output (open), the remaining commands and/or unlock are to be inputted in any order.

With the input of these commands, we can describe the sequencing of the input of these commands as a change of states within the game, where we can identify an initial state and a final/accepting state and determine a finite state automaton (FSA). Here, the FSA's alphabet is the set of all the commands accepted by the game as valid inputs (as described in the outputs of the game) and the language accepted by this is the set of commands mentioned earlier. By applying the conditions for the sequence of inputs for unlocking the door, we can get the following representation of the language:

$\Sigma = \{ L \in \{\text{valid inputs within the game}\}^* \mid L \text{ contains 'unlock' as first string, has the following commands {'unlock', 'mine', 'move', 'craft'} in any order and has 'open' as last string} \}$

With this language, we will proceed to describe the steps required for reaching the secret door. First step, when the game locates us into the game right after entering the correct inputs for starting the game, we begin at the initial state q_0 , from which we move to q_1 after entering the ‘unlock’ command. If we enter ‘unlock’ for a second time right after entering it from q_0 , we will not move from q_1 as no new changes or alterations to the game are noticed, meaning we have a loop for ‘unlock’ in this state. From q_1 and onwards, we will have loops in the states for all repeating commands that we input, meaning that we can only advance in the FSA by inputting the commands exactly once, as long as the condition that ‘unlock’ is the first command entered and ‘open’ is the last entered. For all cases, if the input ‘open’ is entered in intermediate states, the FSA returns the user to q_0 losing all progress (as represented with the green line in the FSA diagram).

Upon opening the secret door the game plots a Dutch flag into the terminal and the player’s inventory gets filled up with all of the block types.

5 Git Collaboration & Version Control

- Repository Link:
https://gitlab.maastrichtuniversity.nl/bcs1110/javacraft/-/tree/group_43
- Branch Details: **group_43**

Git summary: We assigned different tasks of changing the code to different people on the team. Each one split off a different branch in which they worked on their task. After finishing their task, they would commit, push, and merge their branch back to the main group branch, in order to share their progress with the rest. Afterward, they would start a new task and thus a new branch, until the project is finished.

Changes & Conflicts & Problems:

- We had problems at the start understanding how to use git and how to commit/push/pull, but thankfully we had Botond there to help us understand all of this.
- We also had problems at one point when trying to push and pull, in which error messages kept appearing. We solved this by merging the previous branches, then pulling, and finally pushing.
- We had problems when setting up SourceTree, where we needed to create an access token and ssh keys in gitlab. We had many errors so we had to do the process all over again carefully and we made it work.
- Some conflicting branches were very problematic, so we had to copy paste parts of one to the other before effectively pushing them.

6 Extending the Game Code

New additions to the game	Integration into the game
Coal	We added coal as a basic block type in order to later use it as part of a new crafting recipe.
Diamond	We didn't have any use for a second new block type, so we just added the most famous one from Minecraft to fill in the void. We made it rarer than all the other materials in the map, sometimes nonexistent.
Immortal Demon Cow (Refer to appendix for code snippets)	The most complicated and tryhard thing we did. We made a cow represented by a purple "C", that moves around the map in an interval between the player's interactions. By "interacting" with the cow, the player can kill it and obtain meat (while a sad cow sound plays in the background), although the cow will remain alive and keep moving through the map, therefore its title.
Meat	Not exactly a block type, but if you "kill" the cow, it will add meat into your inventory and behave in the same way as other materials.
Crafting table (Refer to appendix for code snippets)	Recipe: 4 wooden planks. It makes possible the crafting of the furnace.
Furnace (Refer to appendix for code snippets)	Recipe: 1 Iron ore, 2 Coals. Only possible to craft it if you have a crafting table in your inventory.
Cooked meat (Refer to appendix for code snippets)	Counts as a crafting recipe, but its kind of the final goal in a way. Recipe: 1 meat. Only possible to make if you have a furnace in your inventory.

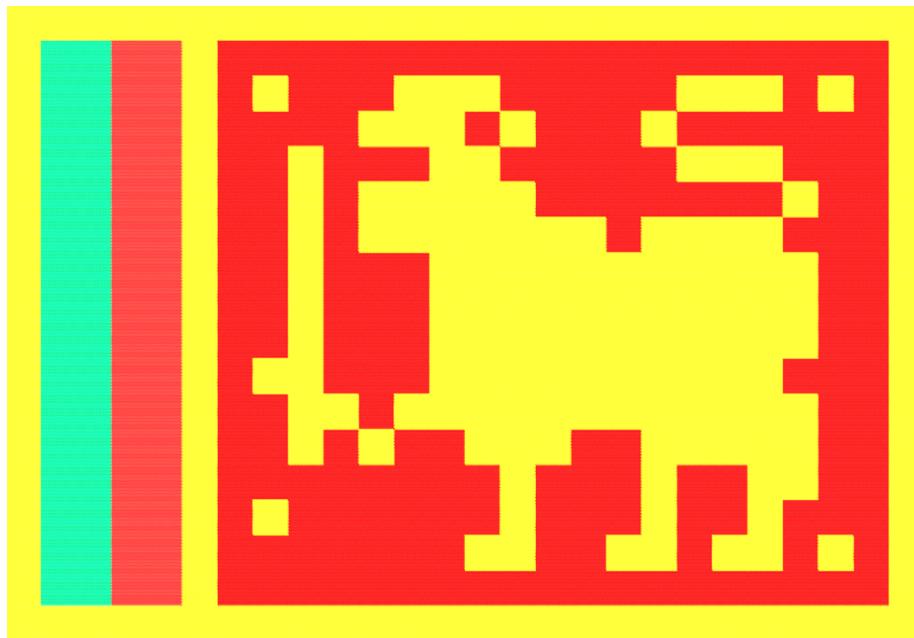
7 Interacting with Flags API (For Final Submission)

In the `getCountryAndQuoteFromServer()` method, we send a request to Ashish's server (with the URL: https://flag.ashish.nl/get_flag), that returns the name of a Country and a Random quote. For the request, we send our group number and group name, as well as the difficulty level that we chose. In our case that was "hard", and the server sent us Sri Lanka as the country name.

The quote is:

"Any code of your own that you haven't looked at for six or more months might as well have been written - someone else - Eagleson's Law".

To render the flag on the grid, we created a 2D int matrix that had numbers from 0 to 3, with each number representing a different color. After using a nested for loop to go through each part of the matrix, we used if-else if statements to print blocks of the color matching the number. We ordered the numbers on the matrix so that when it was printed, it would result in the Sri Lanka flag.



8 Conclusion

Our biggest achievement would be the implementation of the cow in the game, as it also was our biggest challenge to go through. We had many issues such as the cow completely breaking the game, or just spawning on a specific corner of the grid, plus many others. Another achievement would be the flag, as we chose a considerably hard one. And another important achievement would be how our team dynamics were excellent, with no ghost members or people who did nothing. Our biggest challenge next to the cow would have been the use of git, due to its less than friendly usage. After doing this project, we improved in our coding skills, teamwork skills, our familiarity with java, and team management skills considering this was our first ever project.

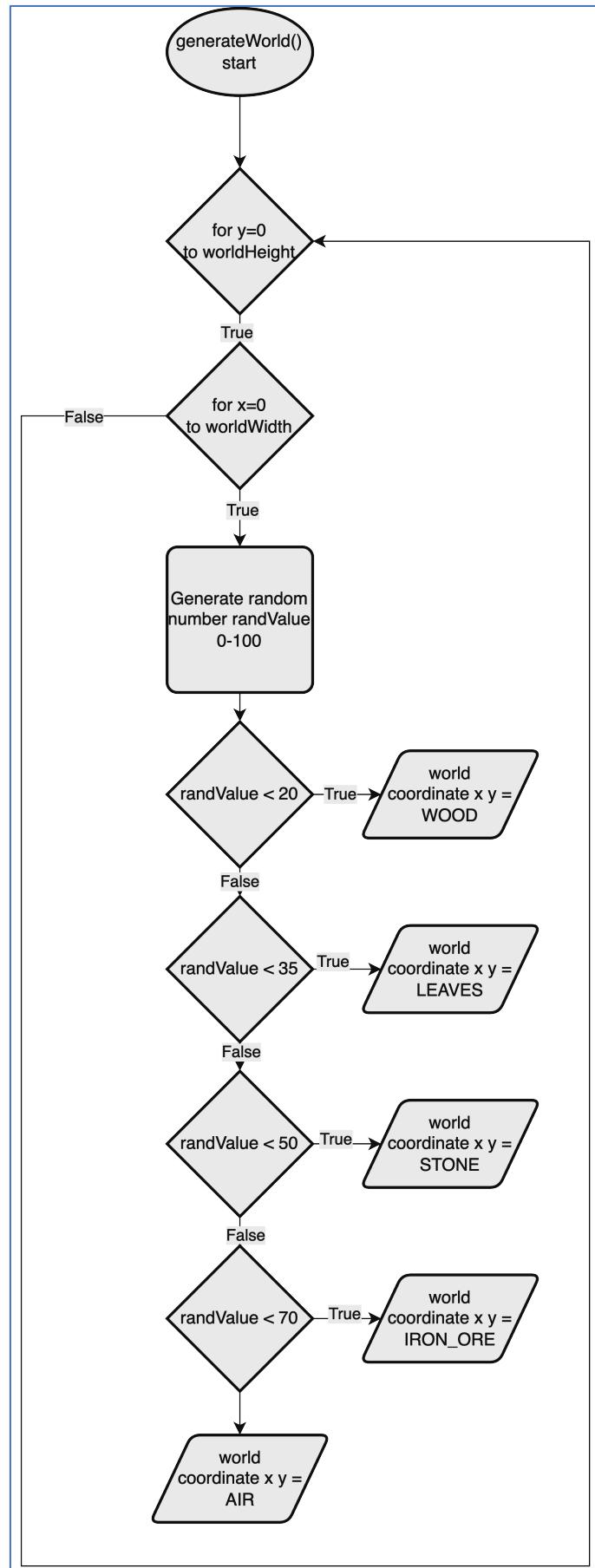
9 References

- Guinto, Susan. *briks. Brik*, <https://cdn.apps.joltteam.com/brikbuild/flag-of-sri-lanka-pixel-art-191320199.pdf>. Accessed 18 Oct. 2023. - Image used for the flag
- Nabizade, M. (2021, June 21). *How to Play Background Music In Java Programs*. Murad Nabizade. Retrieved October 18, 2023, from <https://www.muradnabizade.com/backgroundmusicjava> - Code used for playing the cow sounds.

APPENDIX

Flowcharts & Pseudocode

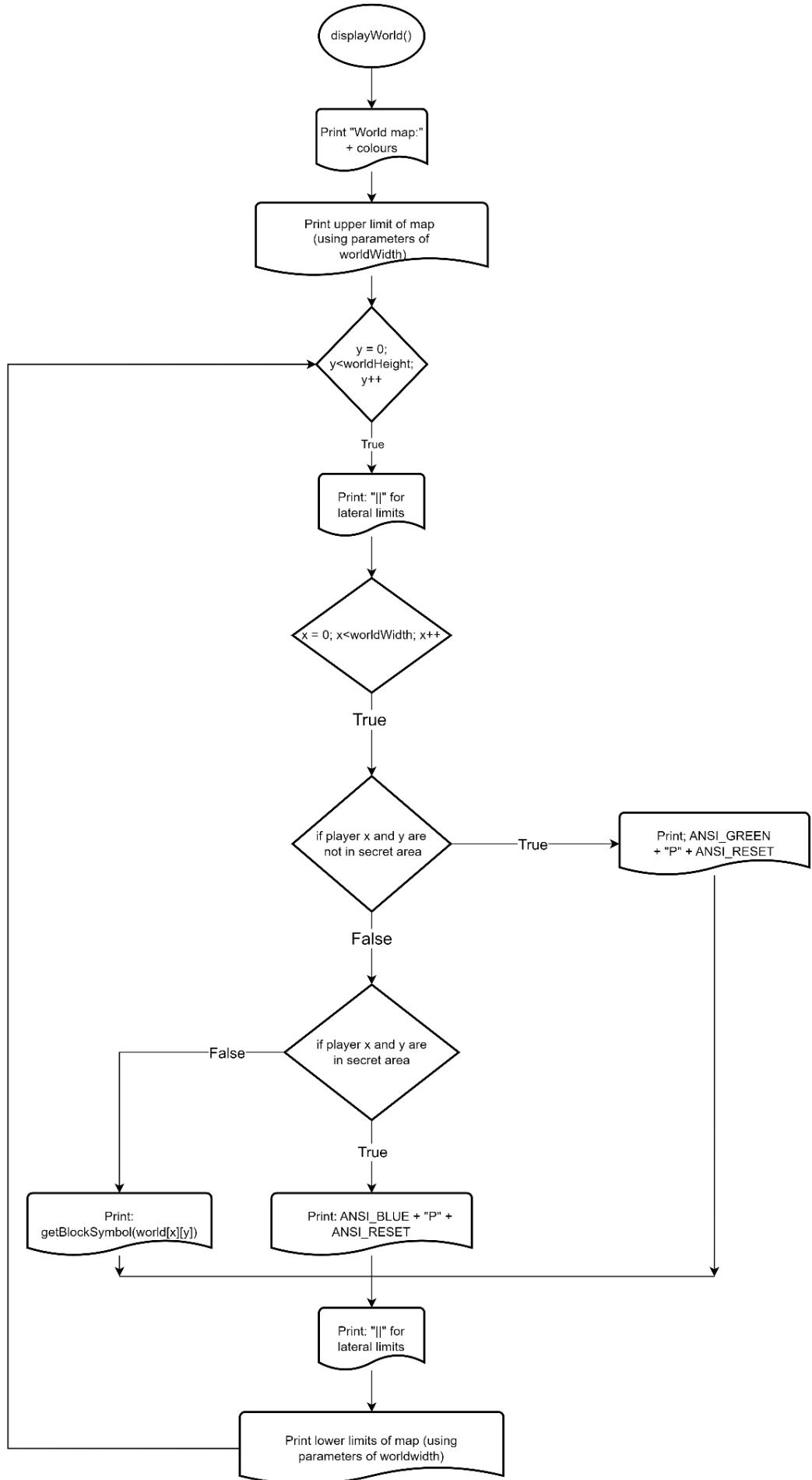
03 – generateWorld



Pseudocodes > ≡ 03-generateWorld

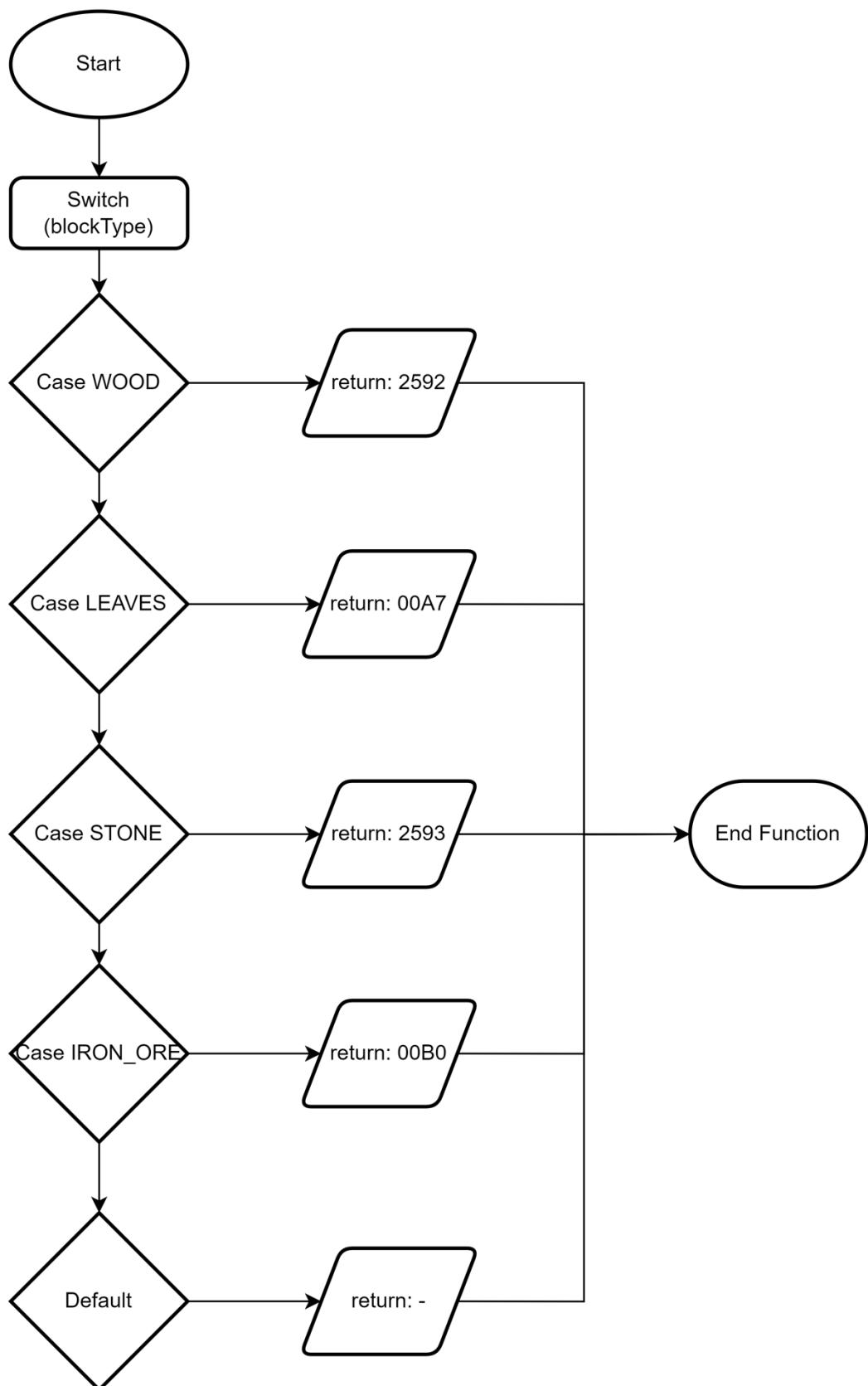
```
1   FUNCTION generateWorld
2       OBJECT Random new rand
3       FOR each y coordinate in wordHeight
4           FOR each x coordinate in worldWidth
5               SET randValue to new random value between 0-100
6               IF randValue is smaller than 20 THEN
7                   SET world xy coordinate to WOOD
8               ELSE IF randValue is smaller than 35 THEN
9                   SET world xy coordinate to LEAVES
10              ELSE IF randValue is smaller than 50 THEN
11                  SET world xy coordinate to STONE
12              ELSE IF randValue is smaller than 70 THEN
13                  SET world xy coordinate to IRON_ORE
14              ELSE
15                  SET world xy coordinate to AIR
```

04 – displayWorld



06 – getBlockChar

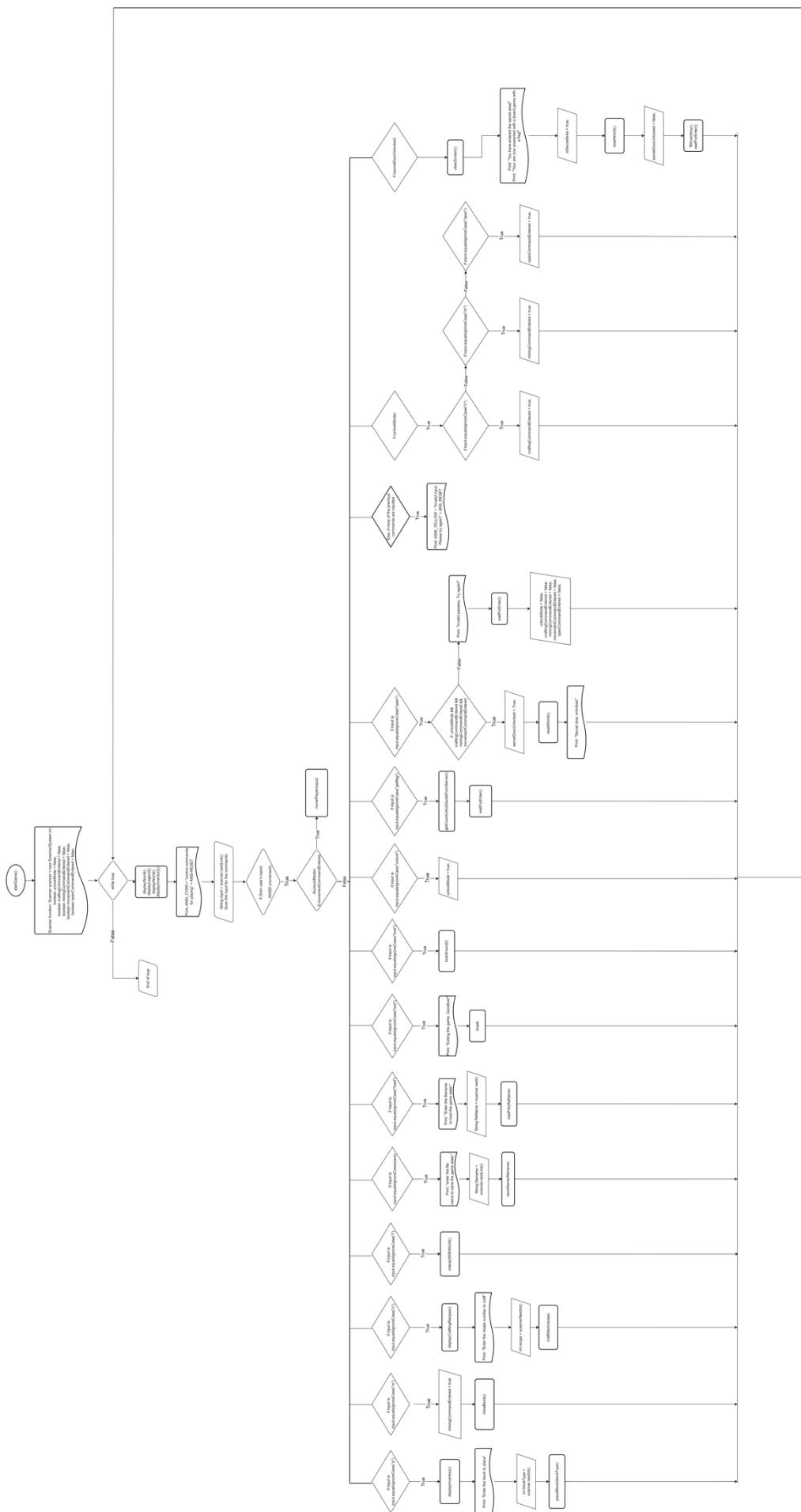
getBlockChar



Pseudocodes > ! 06-getBlockChar

```
1  FUNCTION getBlockChar
2      CASE block type
3          "WOOD":
4              RETURN: 2592
5          "LEAVES":
6              RETURN: 00A7
7          "STONE":
8              RETURN: 2593
9          "IRON_ORE":
10         RETURN: 00B0
11         "DEFAULT":
12         RETURN: nothing
13     ENDCASE
```

07 – startGame



Pseudocodes > 07-startGame

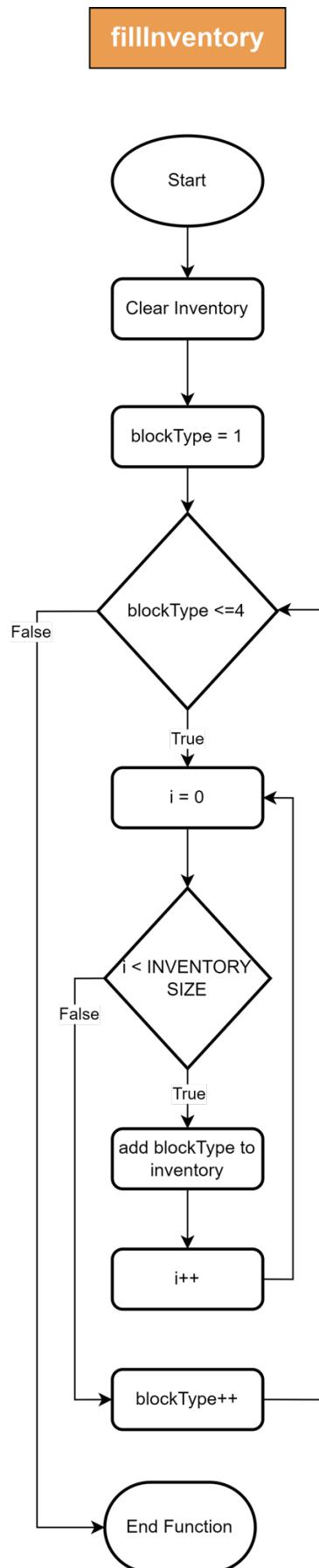
```
1 START GAME
2     scanner is equal to new Scanner(System.in)
3     unlockMode is false
4     craftingCommandEntered is false
5     miningCommandEntered is false
6     movementCommandEntered is false
7     openCommandEntered is false
8 WHILE true
9     clearScreen
10    displayLegend
11    displayWorld
12    displayInventory
13    PRINT "Enter your action: 'WASD': Move, 'M': Mine, 'P': Place, 'C': Craft,
14    'I': Interact, 'Save': Save, 'Load': Load, 'Exit': Quit, 'Unlock': Unlock Secret Door"
15    scanner scans the value for unserinput
16    IF any of these commands are entered (ignoring case) ("w" || "up", "s" || "down", "a" || "left", "d" || "right")
17        IF it is in unlockMode THEN
18            movementCommandEntered is true
19            movePLAYER(Input)
20        ELSE IF INPUT is "m"
21            miningCommandEntered is true
22            mineBlock
23        ELSE IF INPUT is "p"
24            displayInventory
25            PRINT "Enter the block type to place: "
26            blockType is input(integer)
27            placeBlock(blockType)
28        ELSE IF INPUT is "c"
29            displayCraftingRecipes
30            PRINT "Enter the recipe number to craft: "
31            recipe is input(integer)
32            craftItem(recipe)
33        ELSE IF INPUT is "i"
34            interactWithWorld
35        ELSE IF INPUT is "save"
36            PRINT "Enter the file name to save the game state: "
37            fileName is input
38            saveGame(fileName)
39        ELSE IF INPUT is "load"
40            PRINT "Enter the file name to load the game state: "
41            fileName is input
42            loadGame(fileName)
```

```

43     ELSE IF INPUT is "exit"
44         PRINT "Exiting the game. Goodbye!"
45         break
46     ELSE IF INPUT is "look"
47         lookAround
48     ELSE IF INPUT is "unlock"
49         unlockMode is true
50     ELSE IF INPUT is "getFlag"
51         getCountryAndQuoteFromServer
52         waitForEnter
53     ELSE IF INPUT is "open"
54         IF unlockMode && craftingCommandEntered && miningCommandEntered && movementCommandEntered THEN
55             secretDoorUnlocked is true
56             resetWorld
57             PRINT "Secret door unlocked!"
58             waitForEnter
59         ELSE
60             PRINT "Passkey invalid. Try again!"
61             unlockMode is false
62             craftingCommandEntered is false
63             miningCommandEntered is false
64             movementCommandEntered is false
65             openCommandEntered is false
66     ELSE
67         PRINT "Invalid input. Please try again" in yellow
68 IF unlockMode
69     IF INPUT is "c" THEN
70         craftingCommandEntered = true
71     ELSE IF INPUT is "m"
72         miningCommandEntered is true
73     ELSE IF INPUT is "open"
74         openCommandEntered is true
75 IF secretDoorUnlocked
76     clearScreen
77     PRINT "You have entered the secret area!"
78     PRINT "You are not presented with a board game with a flag!"
79     inSecretArea is true
80     resetWorld
81     secretDoorUnlocked is false
82     fillInventory
83     waitForEnter

```

o8 – fillInventory

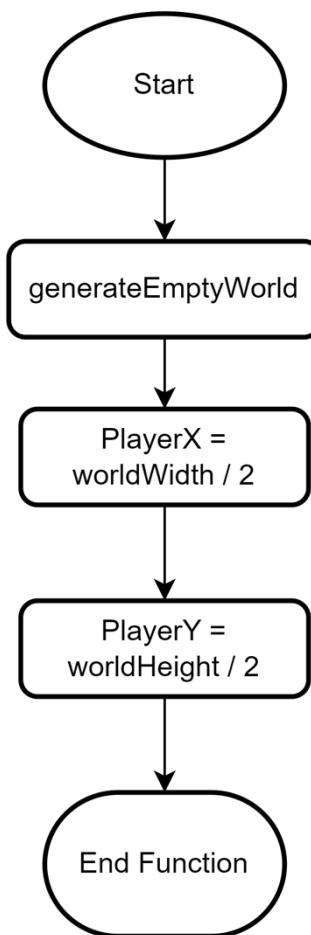


Pseudocodes > Ξ 08-fillInventory

```
1  FUNCTION fillInventory
2      CLEAR inventory
3      FOR each blockType from 1 to 4
4          FOR each in the inventory size
5              ADD block type to the inventory
```

09 – resetWorld

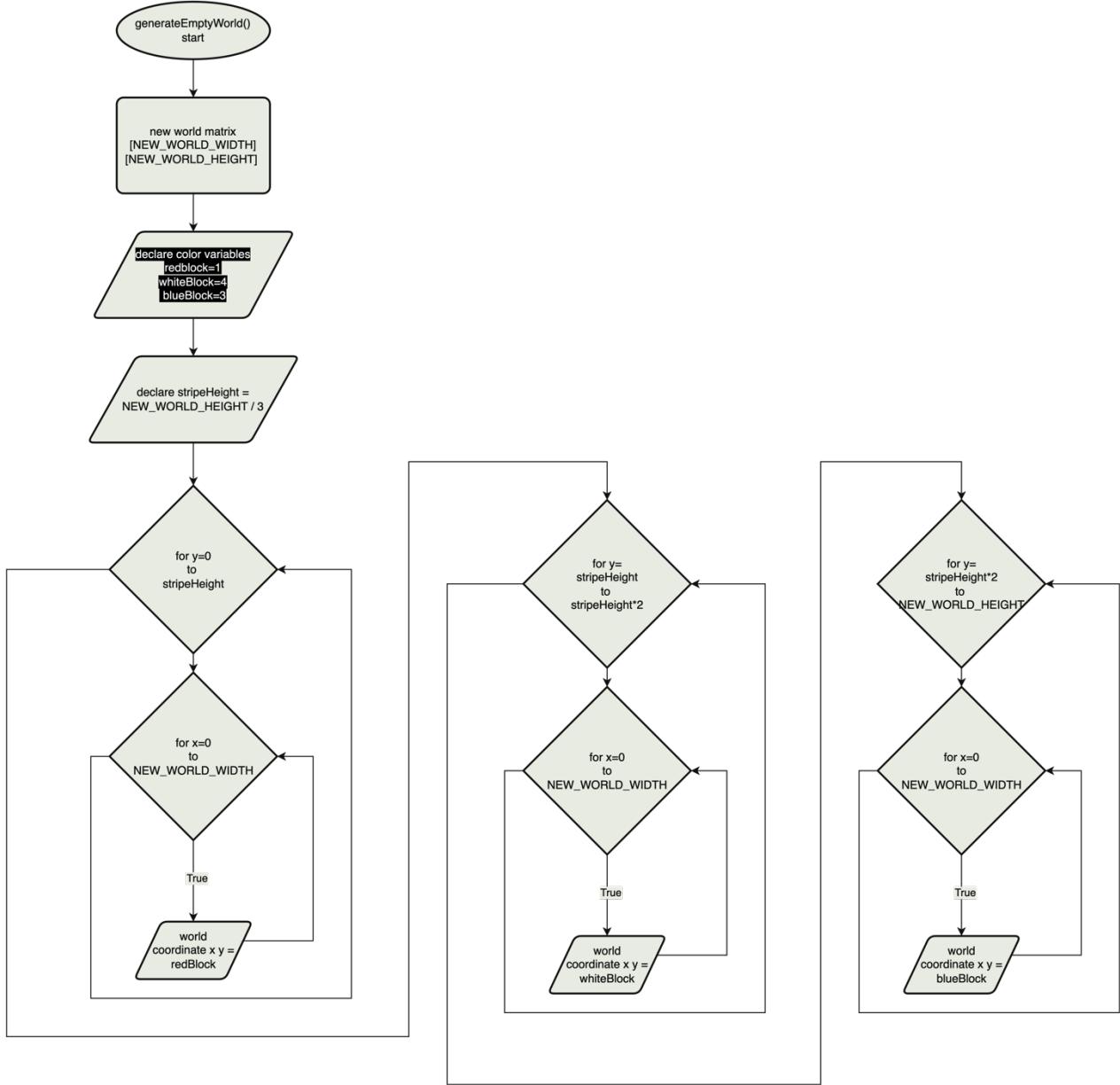
resetWorld



Pseudocodes > 09-resetWorld

```
1  FUNCTION resetWorld
2    GENERATE empty world
3    SET variable playerX to the worldwidth divided by 2
4    SET variable playerY to the worldheight divided by 2
```

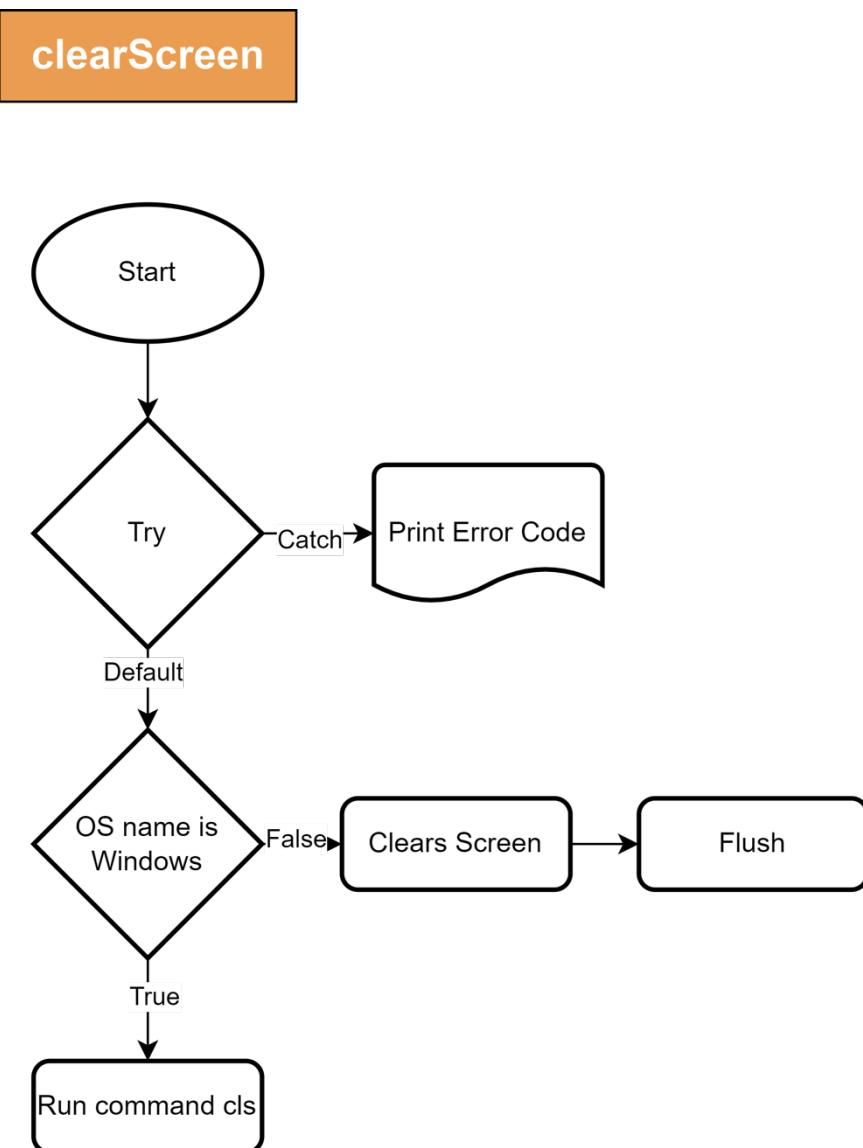
10 – generateEmptyWorld



Pseudocodes > ≡ 10-generateEmptyWorld

```
1 FUNCTION generateEmptyWorld
2     SET world to interger array (NEW_WORLD_WIDTH, NEW_WORLD_HEIGHT)
3     SET redBlock to 1
4     SET whiteBlock to 4
5     SET blueBlock to 3
6     SET stripeHeight to one third of the NEW_WORLD_HEIGHT
7
8     FOR each y coordinate in stripeHeight
9         FOR each x coordinate in NEW_WORLD_WIDTH
10            | SET world xy coordinate to redBlock
11
12     FOR each y coordinate between stripeHeight and 2 times stripeHeight
13         FOR each x coordinate in NEW_WORLD_WIDTH
14             | SET world xy coordinate to whiteBlock
15
16     FOR each y coordinate between 2 times stripeHeight and NEW_WORLD_HEIGHT
17         FOR each x coordinate in NEW_WORLD_WIDTH
18             | SET world xy coordinate to blueBlock
```

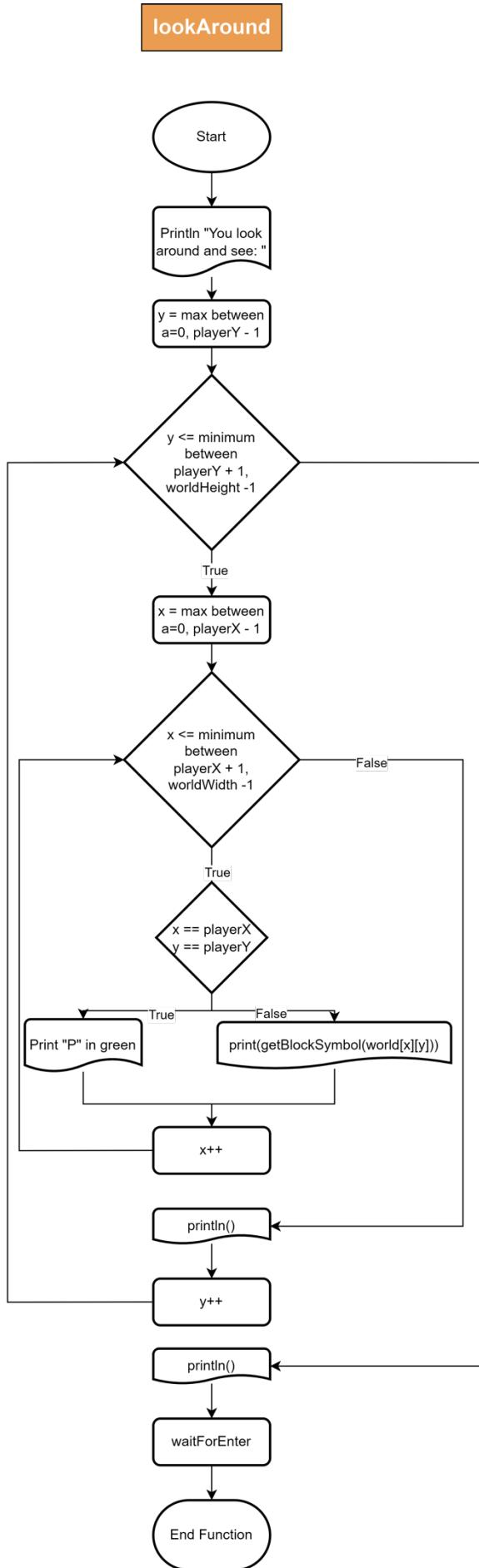
11 – clearScreen



Pseudocodes > ≡ 11-clearScreen

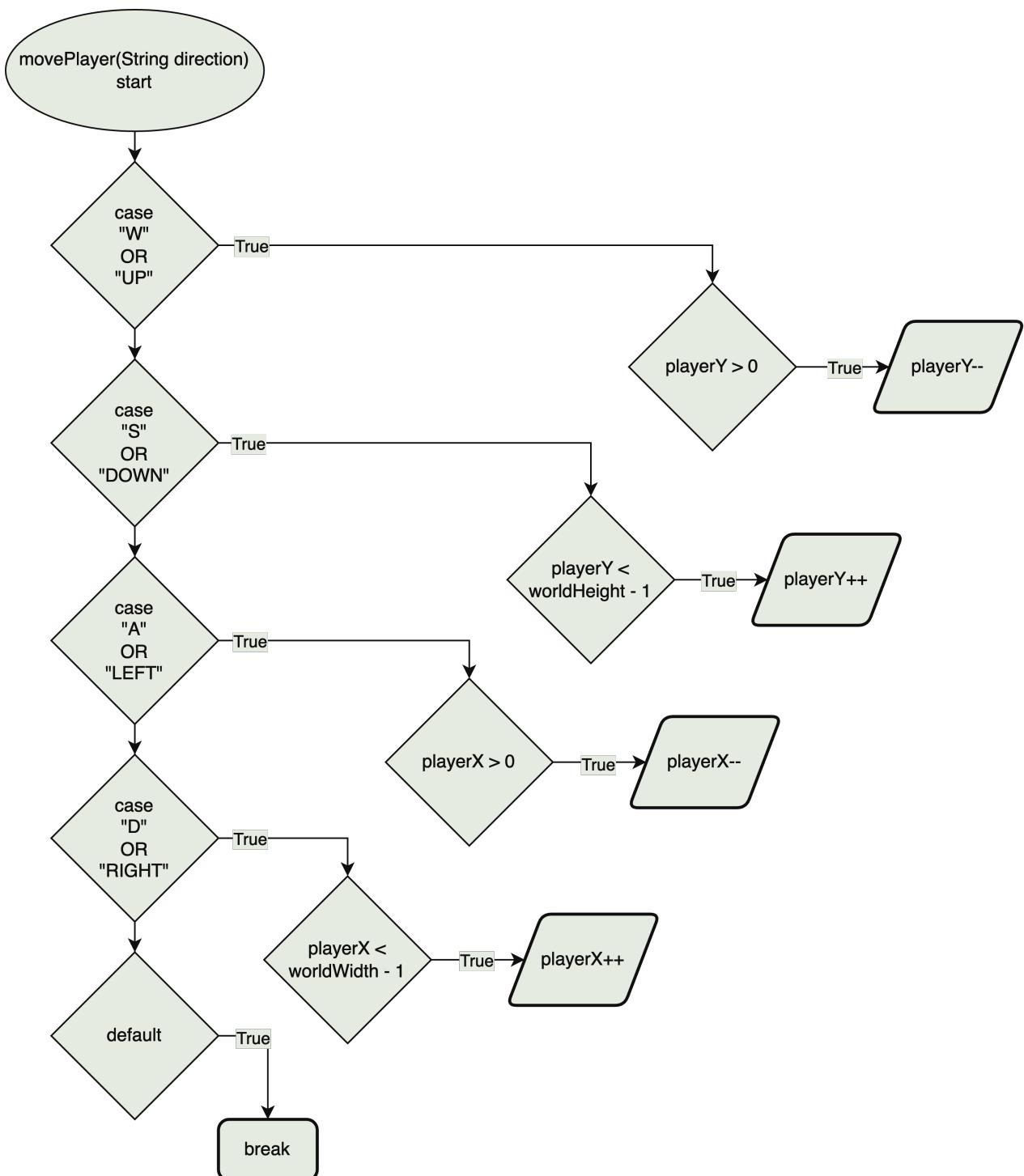
```
1  FUNCTION clearScreen
2      TRY unless an error happens
3          IF the OS name is Windows
4              RUN command cls
5          ELSE
6              CLEAR the screen
7              FLUSH
8      CATCH when an error happens
9          PRINT error code
```

12 – lookAround



```
Pseudocodes > ≡ 12-lookAround
1  FUNCTION lookAround
2    PRINT "You look around and see"
3    FOR each y coordinate that is lower or equal to the minimum between playerY +1, and the worldheight -1
4      FOR each x coordinate that is lower or equal to the minimum between playerX +1, and the worldwidth -1
5        IF the x coordinate equals playerX, and the y coordinate equals playerY
6          PRINT "P" in green
7        ELSE
8          PRINT getBlockSymbol(world[x][y])
9        PRINT a new line
10       PRINT a new line
11       WAIT for enter
```

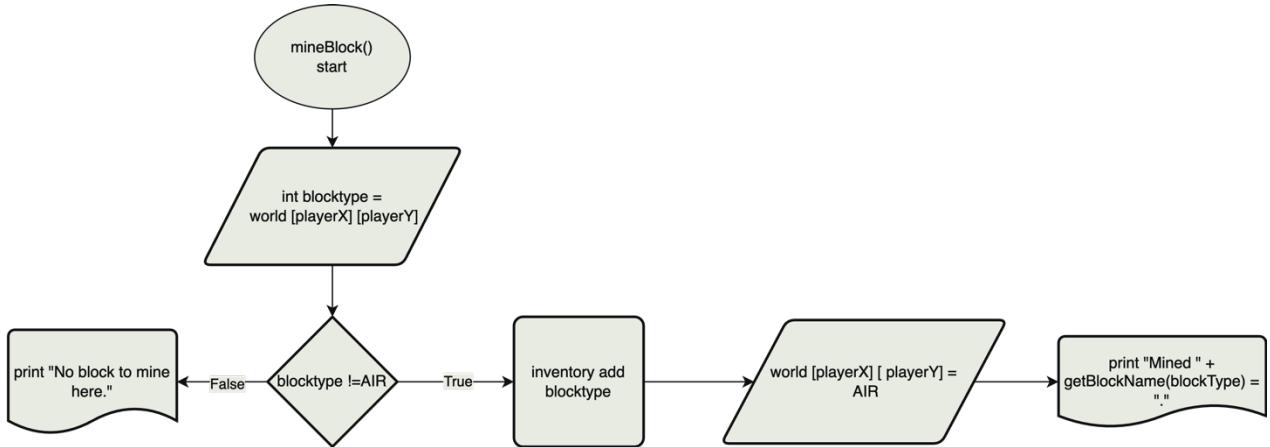
13 – movePlayer



Pseudocodes > ≡ 13-movePlayer

```
1  FUNCTION movePlayer
2    CASE uppercase direction
3      "W" OR "UP":
4          IF playerY is greater than 0 THEN
5              DECREMENT playerY
6      "S" OR "DOWN":
7          IF playerY is smaller than wordHeight-1 THEN
8              INCREMENT playerY
9      "A" OR "LEFT":
10         IF playerX is greater than 0 THEN
11             DECREMENT playerX
12         "D" OR "RIGHT":
13             IF playerX is smaller than worldWidth-1 THEN
14                 INCREMENT playerX
15             DEFAULT:
16             ENDCASE
```

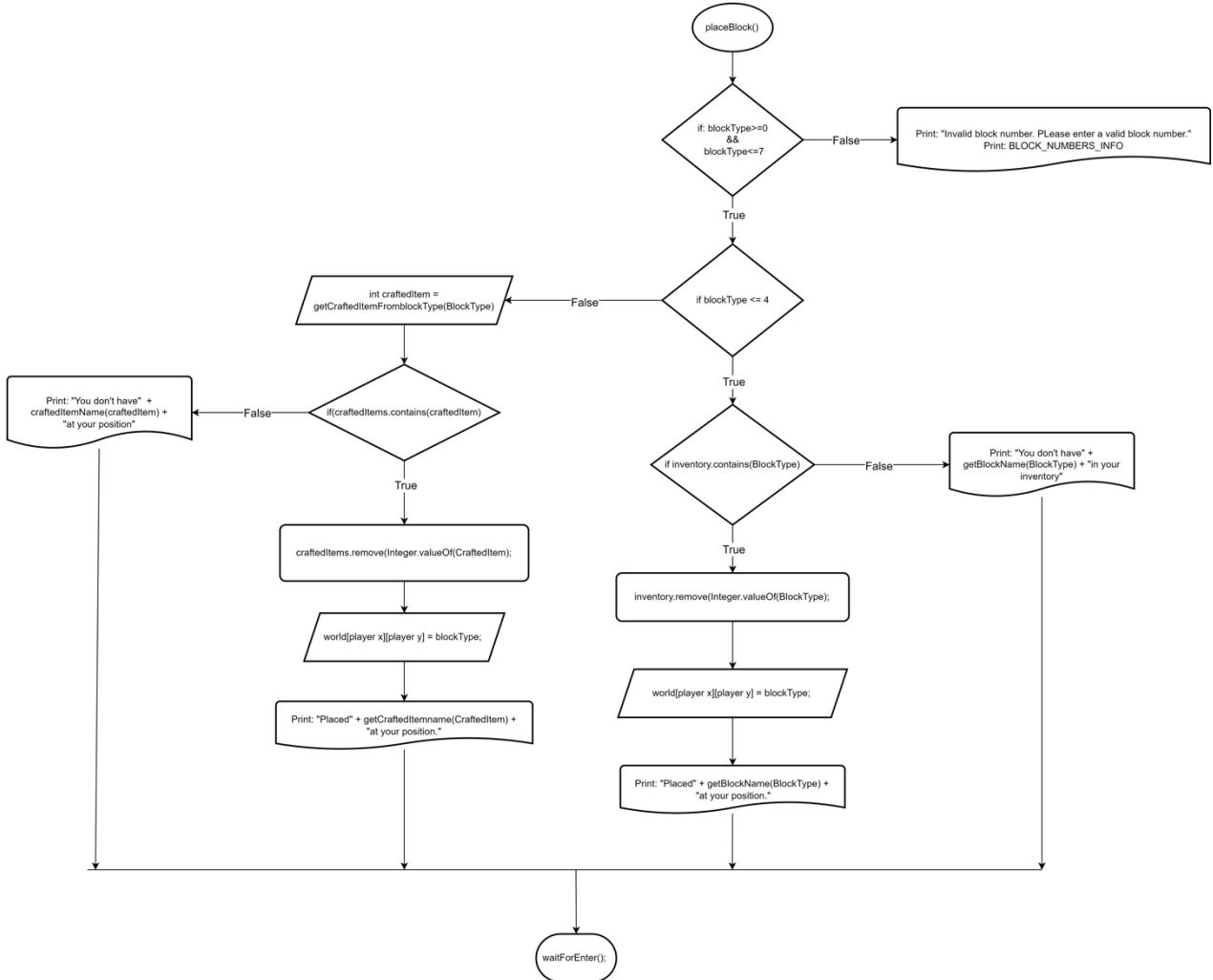
14 – mineBlock



Pseudocodes > ≡ 14-mineBlock

```
1  ↘ FUNCTION mineBlock
2          SET blockType to world(playerX, playerY)
3  ↘      IF blockType is not AIR THEN
4          ADD inventory blockType
5          SET world(playerX,playerY) to AIR
6          PRINT "Mined" GET blockname "."
7  ↘      ELSE
8          PRINT "No block to mine here."
```

15 – placeBlock



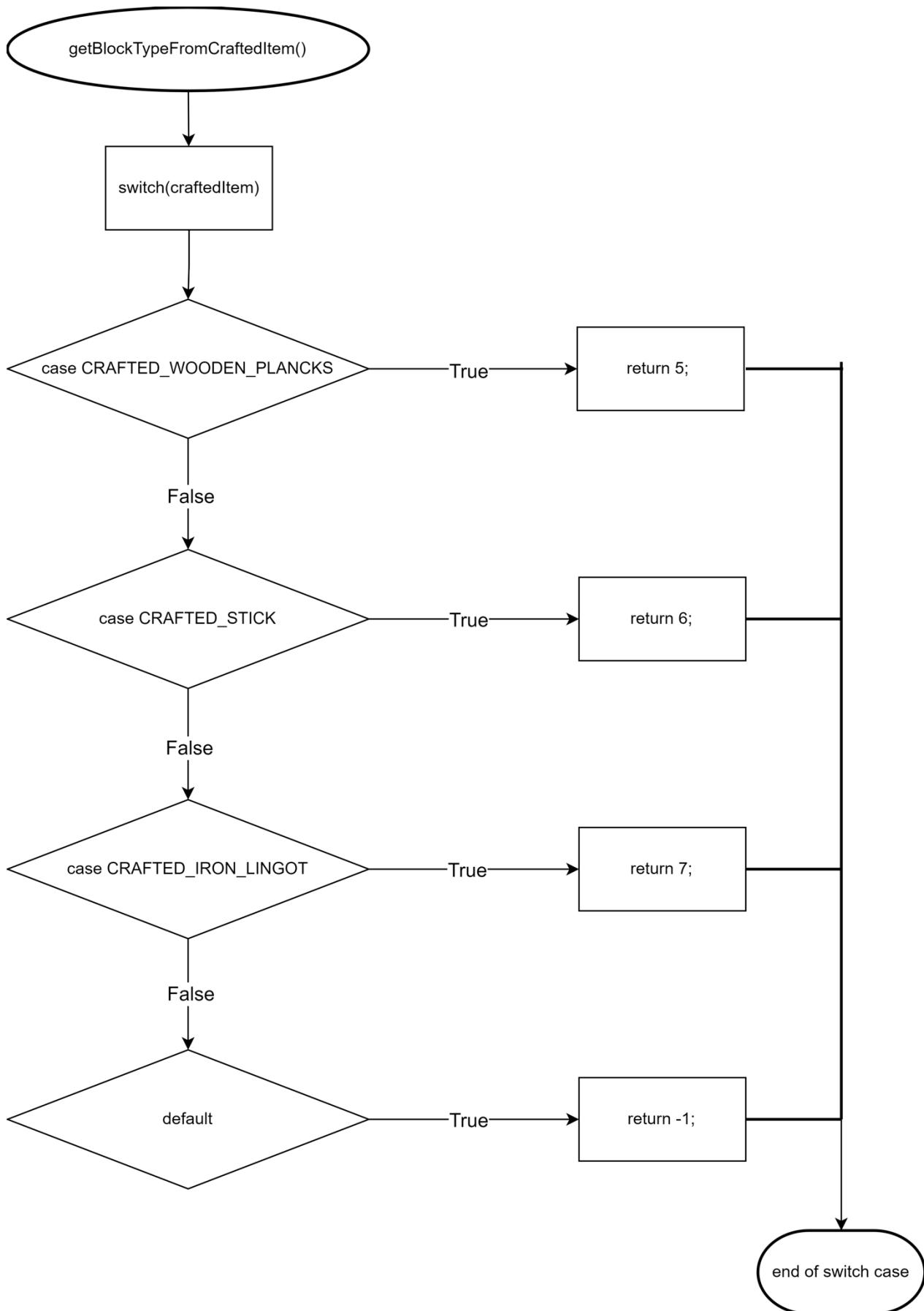
Pseudocodes > 15-placeBlock

```

1  PLACE BLOCK
2  IF blockType is greater or equal to 0 and blockType is also smaller or equal to 7
3      IF blockType is smaller or equal to 4
4          IF inventory contains blockType
5              Remove an integer value of blockType in inventory
6              world[player X][player Y] is equal to blockType
7              PRINT "Placed " + getBlockName(blockType) + " at your position."
8          ELSE
9              PRINT "You don't have " + getBlockName(blockType) + " in your inventory."
10     ELSE
11         craftedItem equals to getCraftedItemFromBlockType(blockType)
12         IF craftedItems contains craftedItem THEN
13             Remove an integer value in craftedItem
14             world[playerX][playerY] is equal to blockType
15             PRINT "Placed " + getCraftedItemName(craftedItem) + " at your position."
16         ELSE
17             PRINT "You don't have " + getCraftedItemName(craftedItem) + " in your crafted items."
18     ELSE
19         PRINT "Invalid block number. Please enter a valid block number."
20         PRINT BLOCK_NUMBERS_INFO
21     waitForEnter()

```

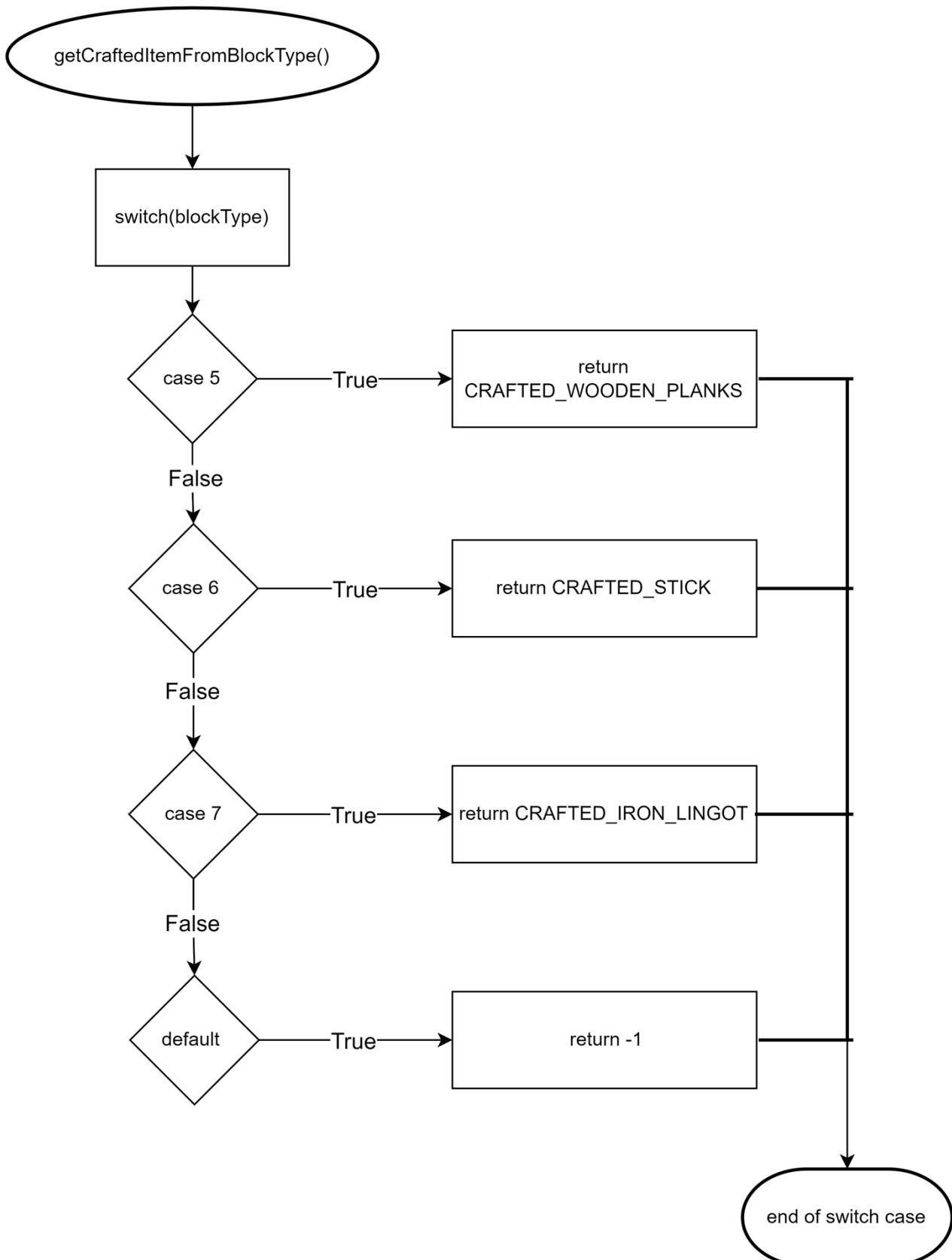
16 – getBlockTypeFromCraftedItem



Pseudocodes > ≡ 16-getBlockTypeFromCraftedItem

```
1   GET BLOCK TYPE FROM CRAFTED ITEM
2       SWITCH craftedItem
3           CASE CRAFTED_WOODEN_PLANKS:
4               return 5
5           CASE CRAFTED_STICK:
6               return 6
7           CASE CRAFTED_IRON_INGOT:
8               return 7
9           DEFAULT:
10              return -1
```

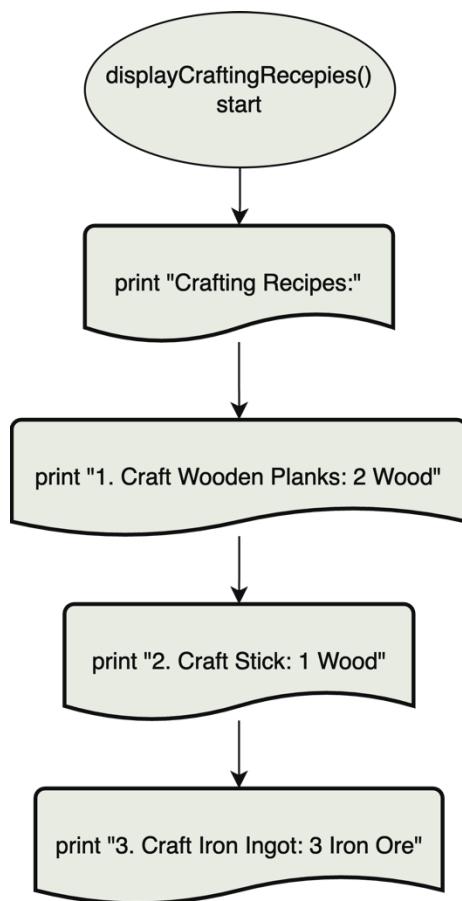
17 – getCraftedItemFromBlockType



Pseudocodes > ≡ 17-getCraftedItemFromBlockType

```
1   GET CRAFTED ITEM FROM BLOCK TYPE
2       SWITCH blockType
3           CASE 5
4               return CRAFTED_WOODEN_PLANKS
5           CASE 6
6               return CRAFTED_STICK
7           CASE 7
8               return CRAFTED_IRON_INGOT
9       DEFAULT
10      return -1
```

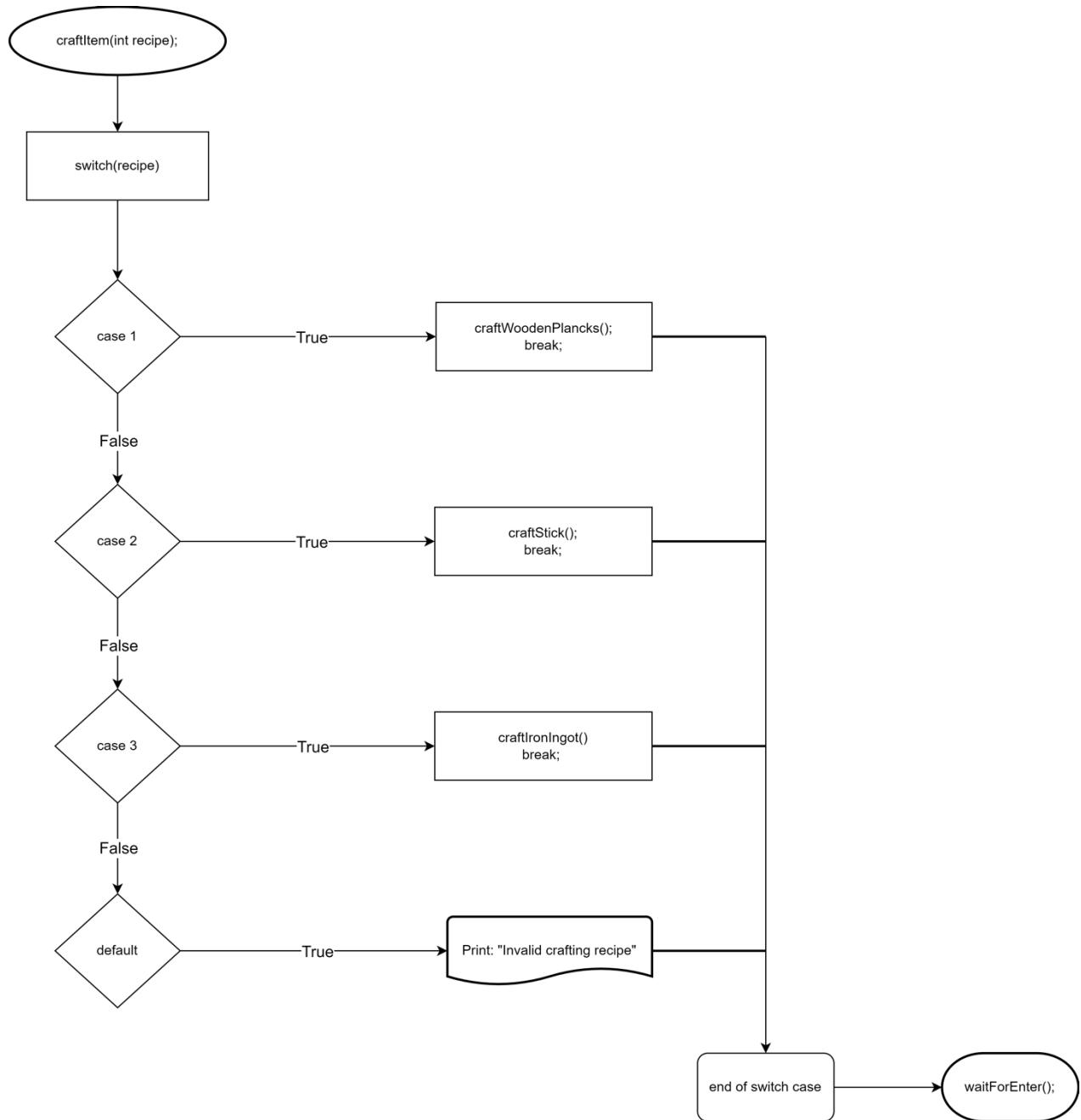
18 – displayCraftingRecepies



Pseudocodes > 18-displayCraftingRecipes

```
1  FUNCTION displayCraftingRecipes
2      PRINT "Crafting Recipes:"
3      PRINT "1. Craft Wooden Planks: 2 Wood"
4      PRINT "2. Craft Stick: 1 Wood"
5      PRINT "3. Craft Iron Ingot: 3 Iron Ore"
```

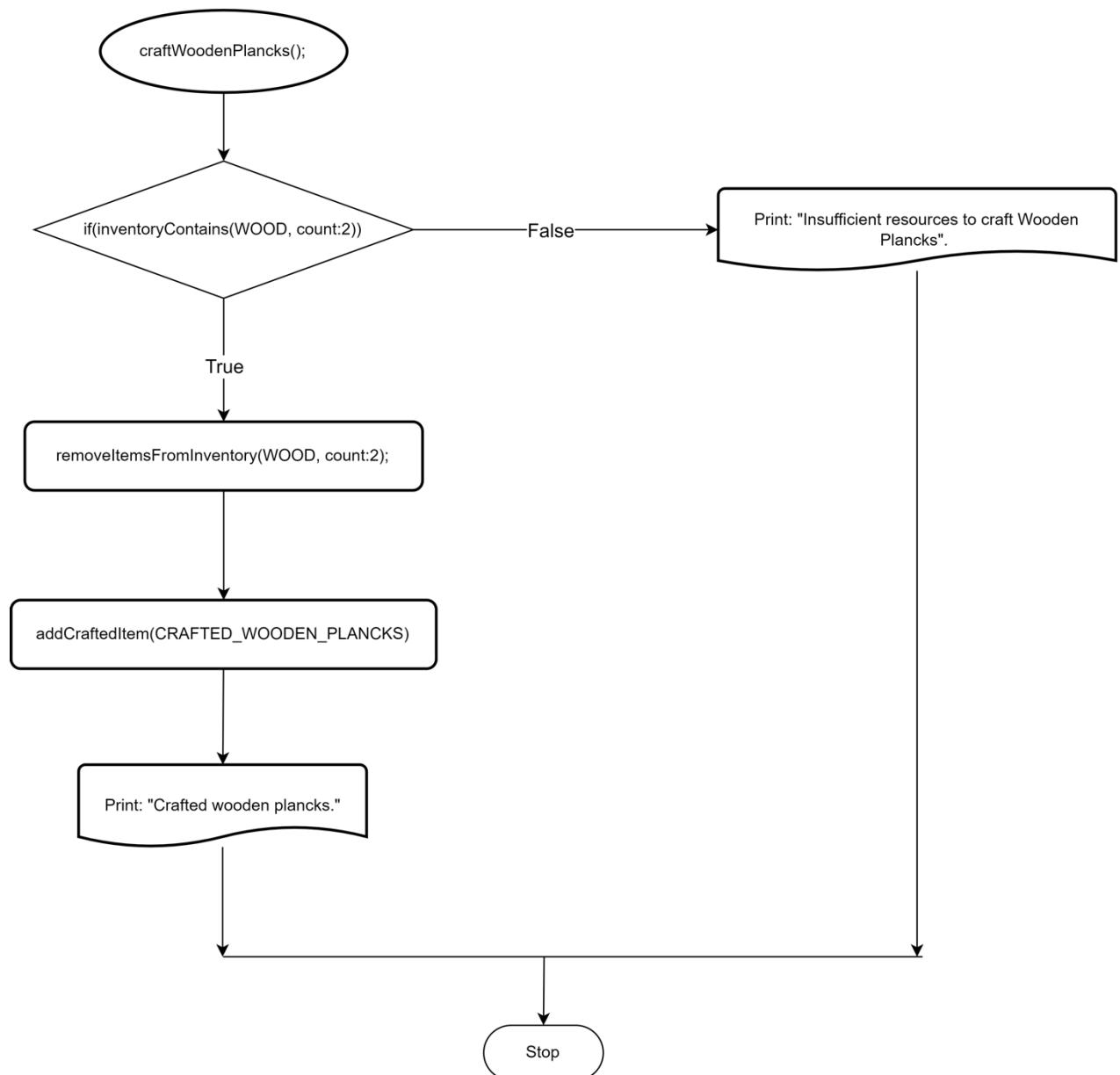
19 – craftItem



Pseudocodes > 19-craftItem

```
1 CRAFT ITEM
2     SWITCH recipe
3         CASE 1
4             craftWoodenPlanks
5             break
6         CASE 2
7             craftStick
8             break
9         CASE 3
10            craftIronIngot
11            break
12        DEFAULT
13            PRINT "Invalid recipe number."
14    waitForEnter
```

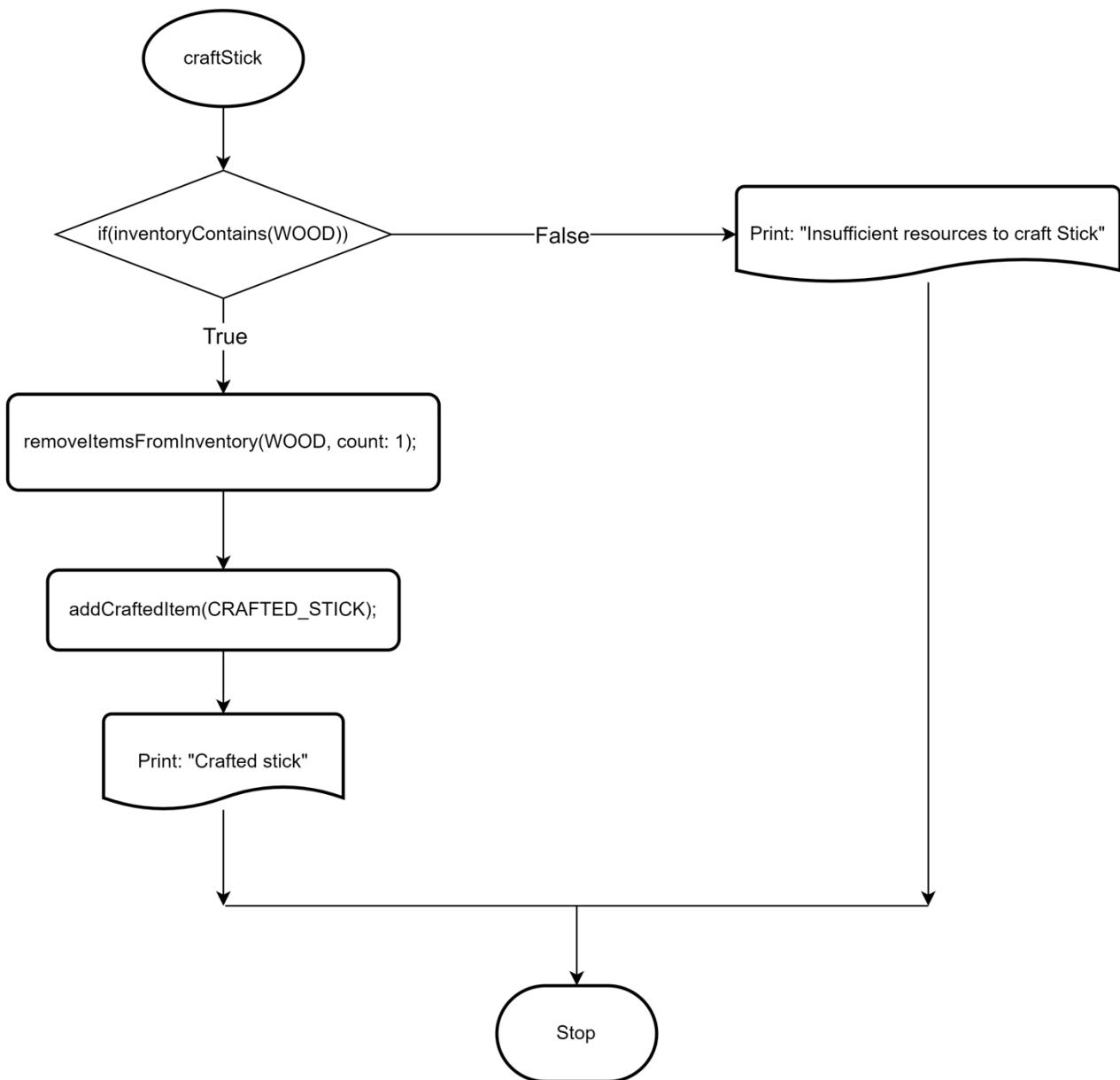
20 – craftWoodenPlanks



Pseudocodes > 20-craftWoodenPlanks

```
1 CRAFT WOODEN PLANKS
2 IF inventory contains wood (2 units min) THEN
3     Remove 2 units of wood from inventory
4     addCraftedItem(CRAFTED_WOODEN_PLANKS)
5     PRINT "Crafted wooden planks"
6 ELSE
7     PRINT "Insufficient resources to craft Wooden Plancks. "
```

21 – craftStick



Pseudocodes > ≡ 21-craftStick

```
1 CRAFT STICK
2     IF inventory contains wood THEN
3         Remove 1 unit of wood from inventory
4         addCraftedItem(CRAFTED_STICK)
5         PRINT "Crafted Stick."
6     ELSE
7         PRINT "Insufficient resources to craft Stick."
```

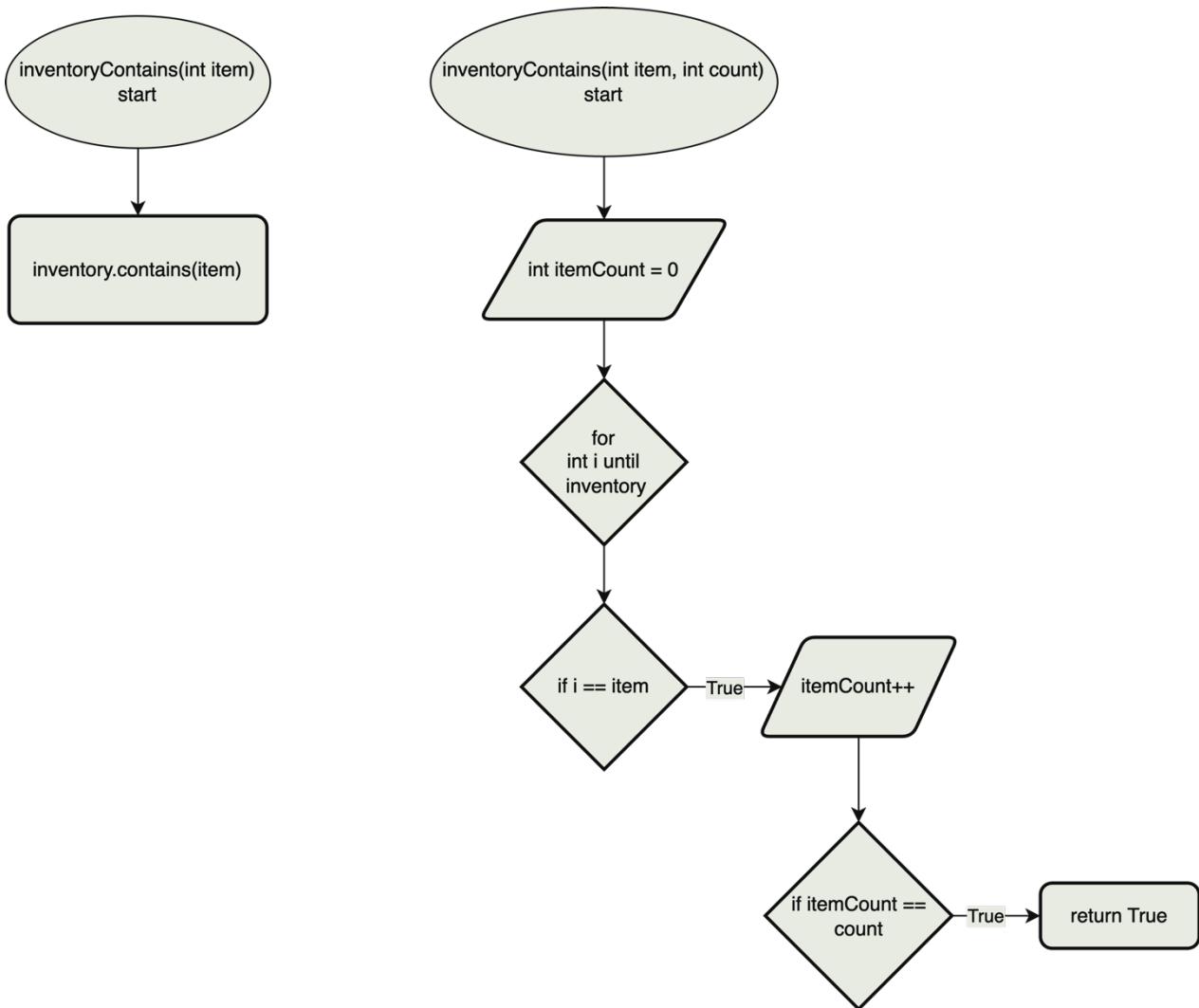
22 – craftIronIngot



Pseudocodes > 22-craftIronIngot

```
1 CRAFT_IRON_INGOT
2     IF inventory contains iron ore (3 units min) THEN
3         Remove 3 units of iron ore from inventory
4         addCraftedItem(CRAFTED_IRON_INGOT)
5         PRINT "Crafted Iron Ingot."
6     ELSE
7         PRINT "Insufficient resources to craft Iron Ingot"
```

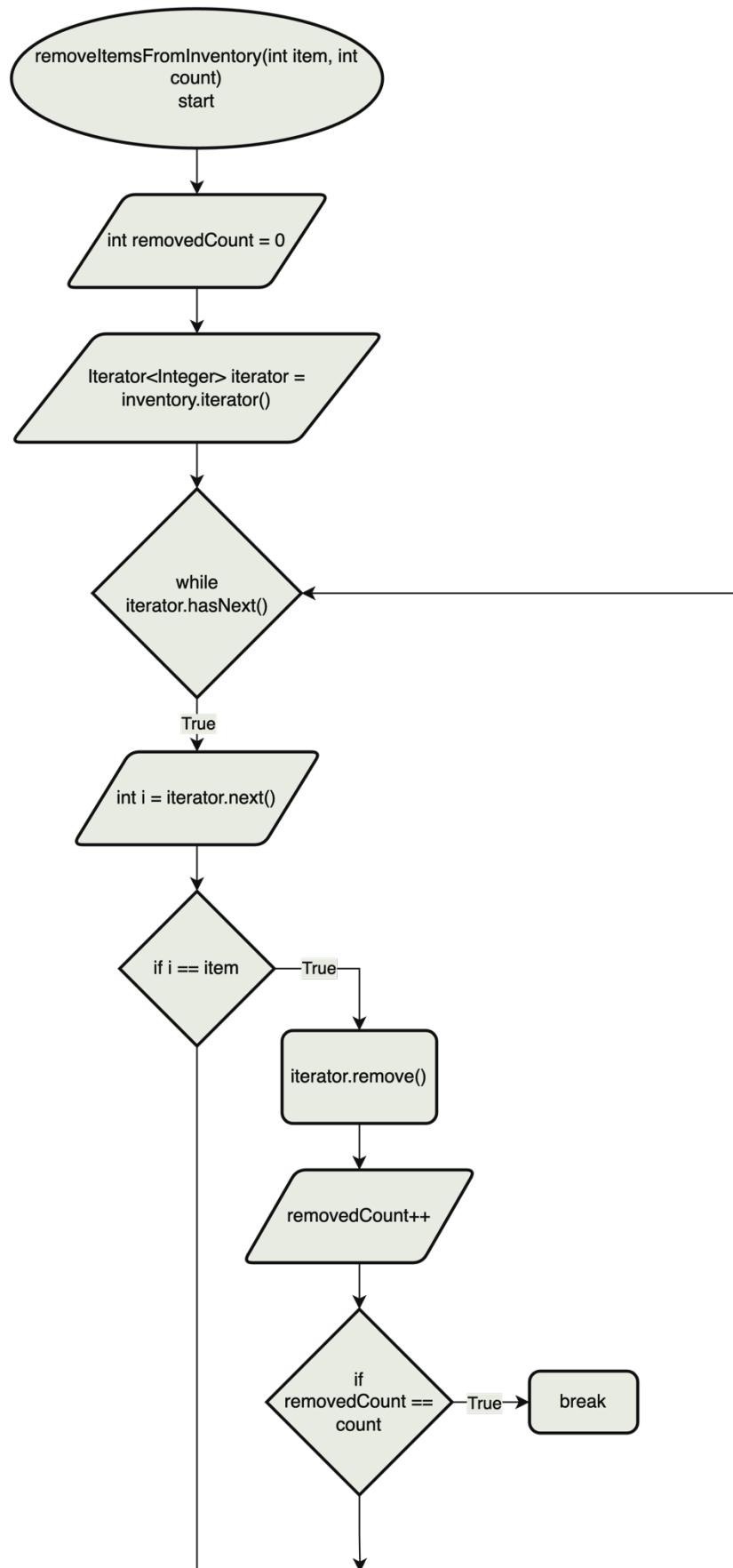
23-24 – inventoryContains



Pseudocodes > 23-24-inventoryContains

```
1  FUNCTION inventoryContains(integer item)
2      RETURN wheter inventory contains(item)
3
4  FUNCTION inventoryContains(integer item, integer count)
5      SET itemCount to 0
6      FOR each i in inventory
7          IF i equals to item THEN
8              INCREMENT itemCount
9              IF itemCount equals count THEN
10                 RETURN true
11             RETURN false
```

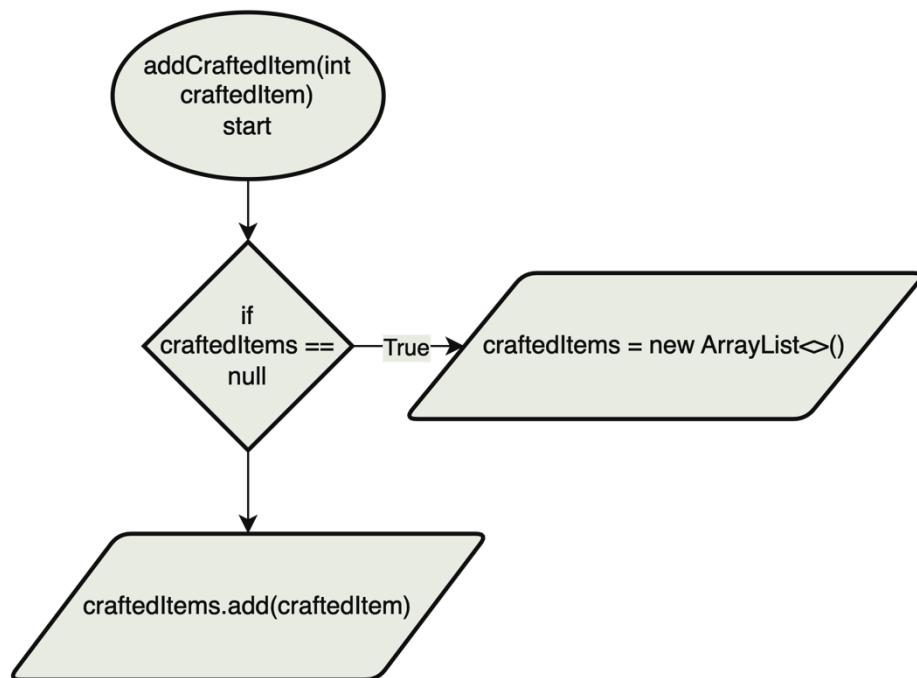
25 – removeItemsFromInventory



Pseudocodes > ≡ 25-removeItemsFromInventory

```
1   FUNCTION removeItemsFromInventory
2     SET removedCount to 0
3     NEW iterator inventory.iterator
4     WHILE iterator has next item
5       SET i to next iterator item
6       IF i equals to item
7         REMOVE iterator item
8         INCREMENT removedCount
9         IF removedCount equals to count THEN
10           BREAK
```

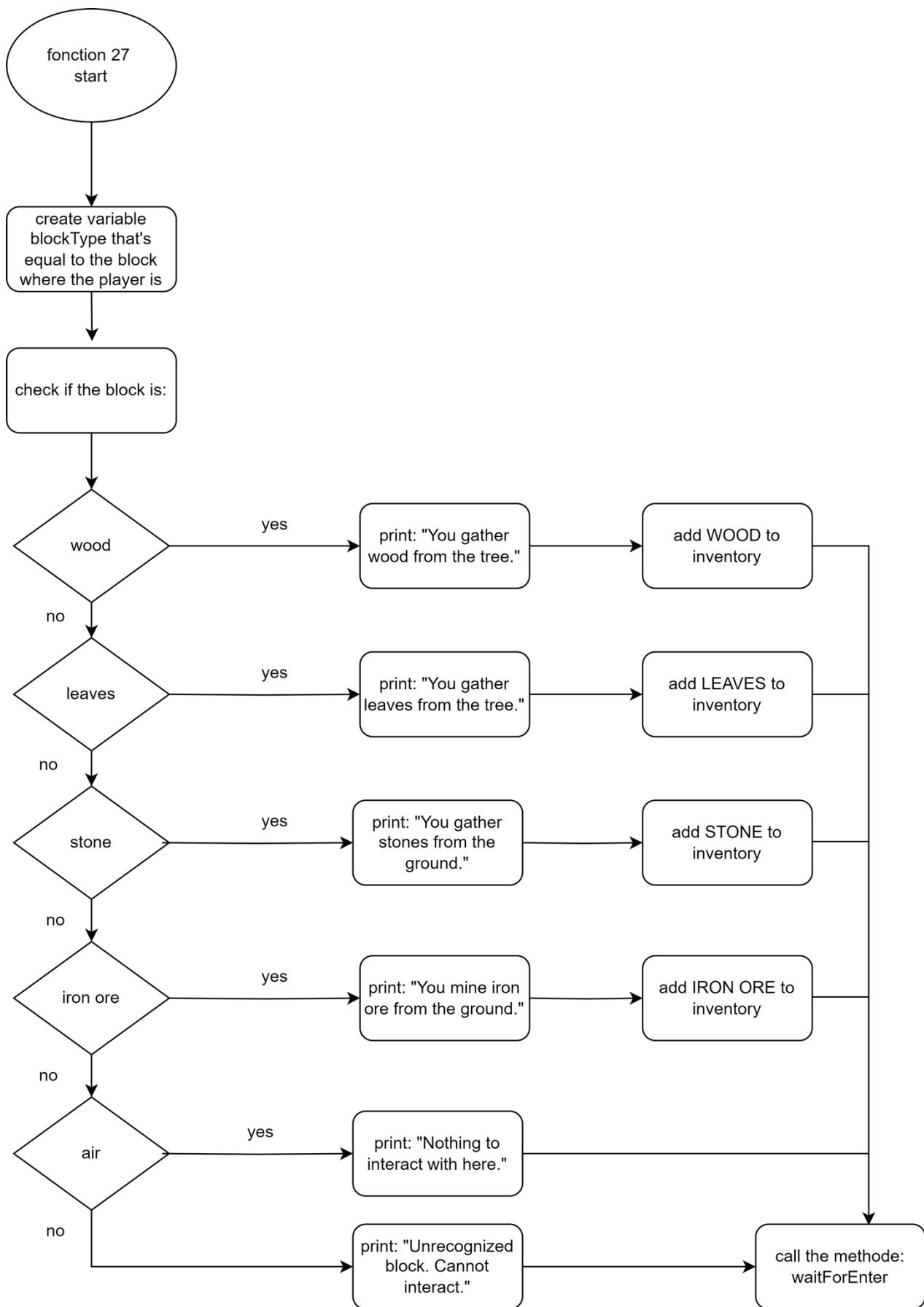
26 – addCraftedItem



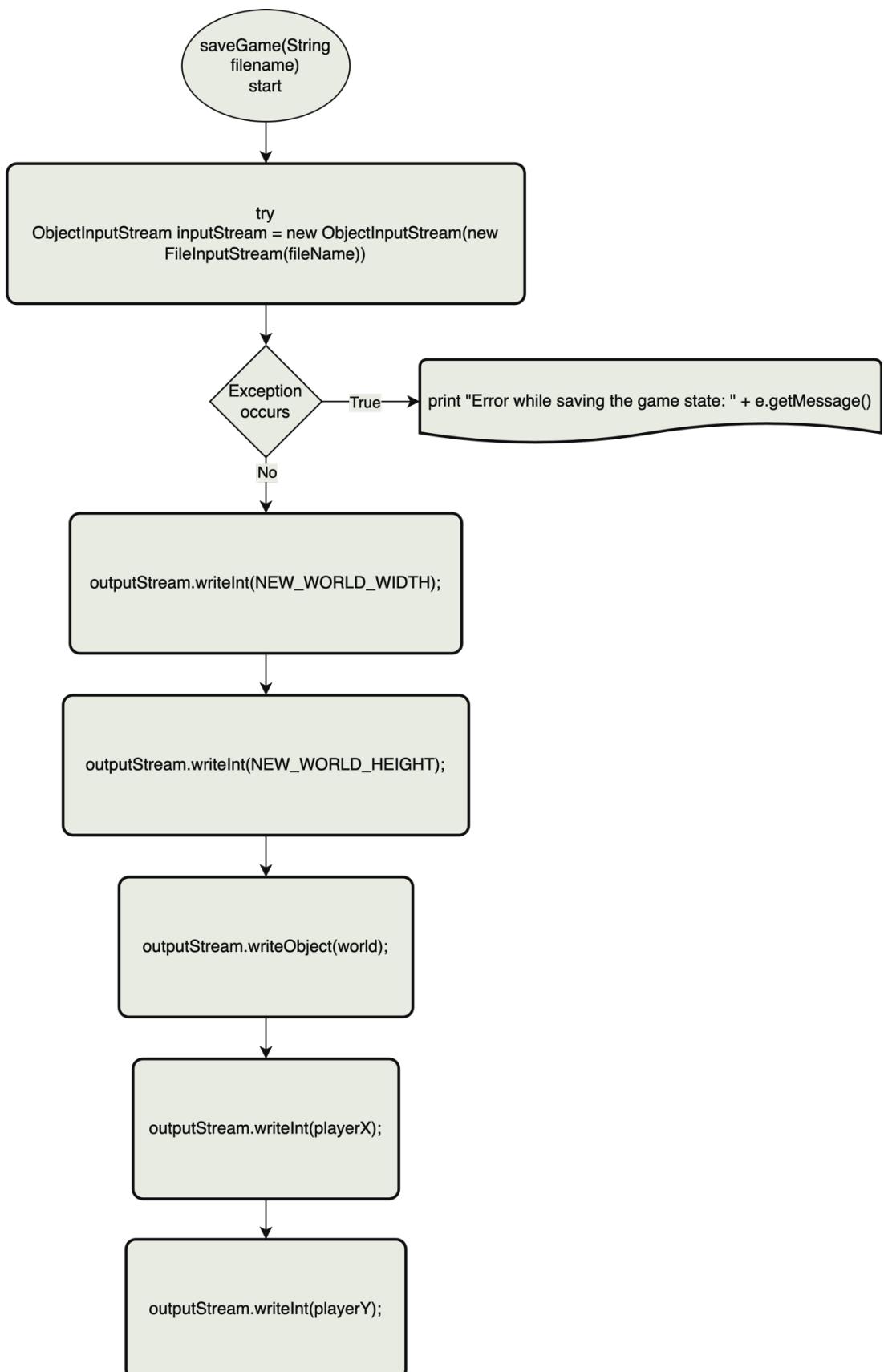
Pseudocodes > 26-addCraftedItem

```
1  FUNCTION addCraftedItem
2    IF craftedItems equals to null THEN
3      NEW ArrayList craftedItems
4      ADD craftedItem to craftedItems
```

27 – interactWithWorld

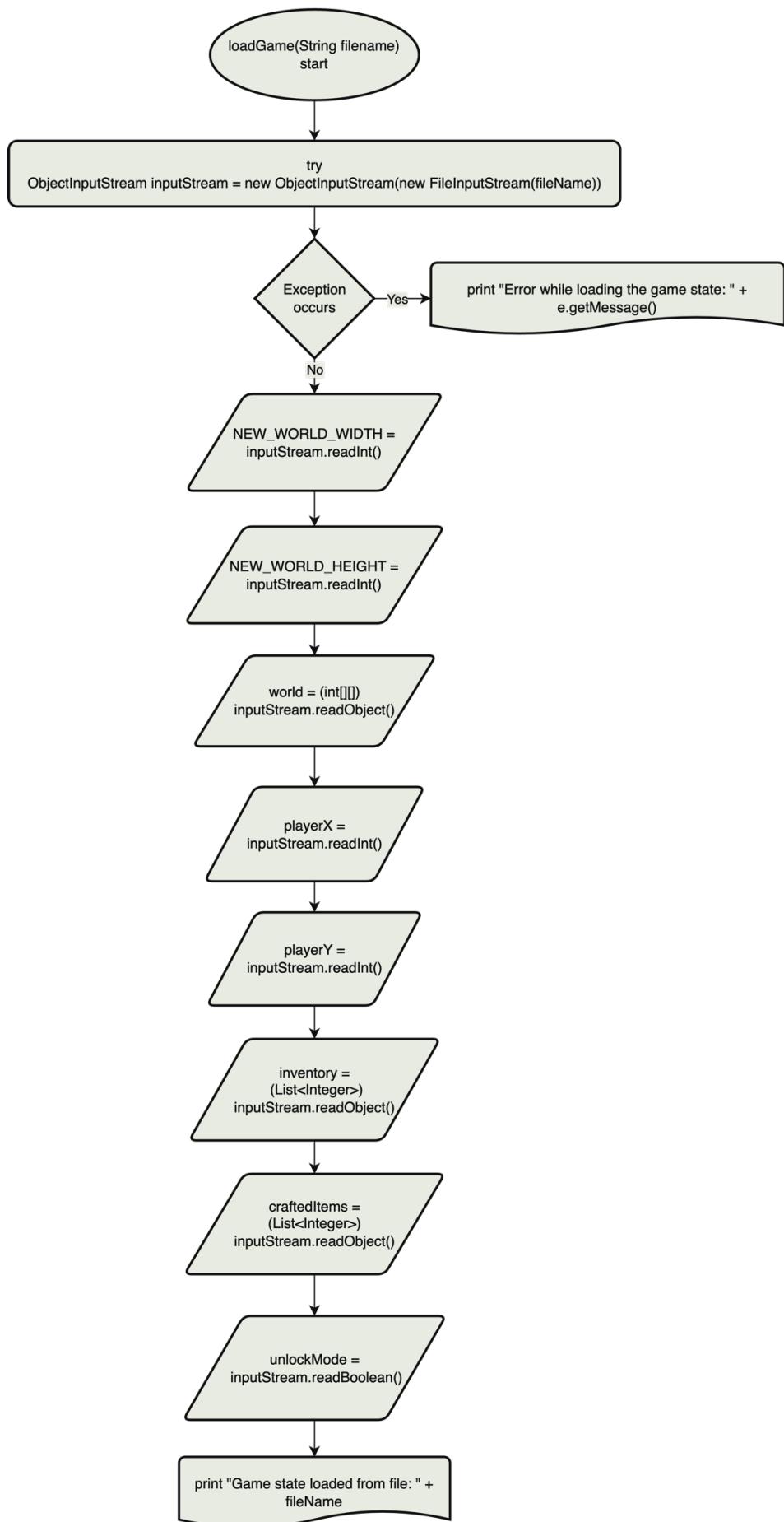


28 – saveGame



```
FUNCTION saveGame(string fileName)
    TRY NEW ObjectOutputStream(NEW FileOutputStream(fileName)) outputStream
        WRITE NEW_WORLD_WIDTH to outputStream
        WRITE NEW_WORLD_HEIGHT to outputStream
        WRITE world to outputStream
        WRITE playerX to outputStream
        WRITE playerY to outputStream
        WRITE inventory to outputStream
        WRITE craftedItems to outputStream
        WRITE unlockMode to outputStream
        PRINT "Game state saved to file: " + fileName
    CATCH
        PRINT "Error while saving the game state: " + error message
    waitForEnter()
```

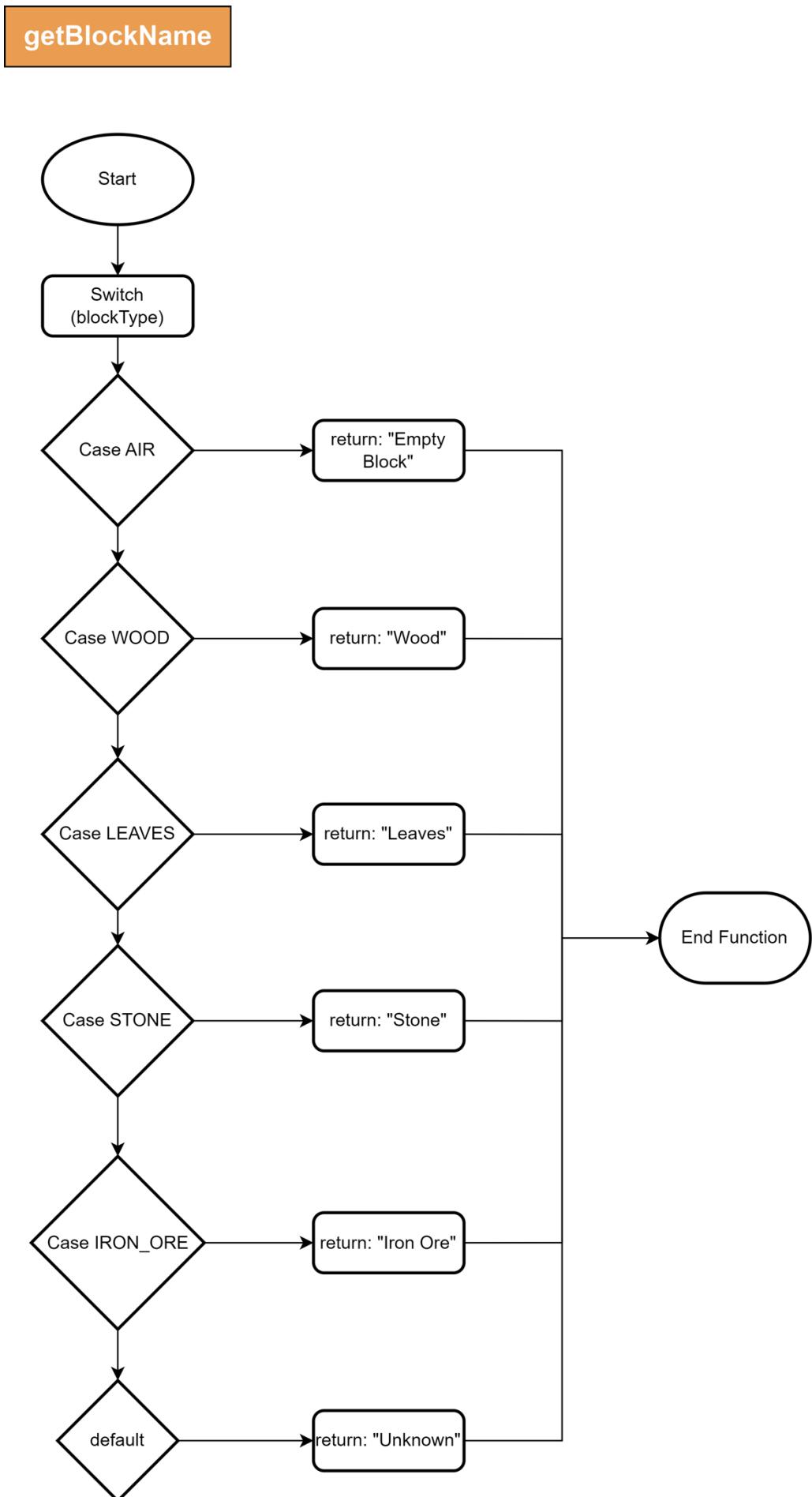
29 – loadGame



Pseudocodes > 29-loadGame

```
1  FUNCTION loadGame(string fileName)
2      TRY NEW ObjectInputStream(NEW FileInputStream(fileName)) inputStream
3          SET NEW_WORLD_WIDTH to READ inputStream
4          SET NEW_WORLD_HEIGHT to READ inputStream
5          SET world to READ inputStream
6          SET playerX to READ inputStream
7          SET playerY to READ inputStream
8          SET inventory to READ inputStream
9          SET craftedItems to READ inputStream
10         SET unlockMode to REAF inputStream
11         PRINT "Game state loaded from file: " + fileName
12     CATCH
13         PRINT "Error while loading the game state: " + error message
14    waitForEnter()
```

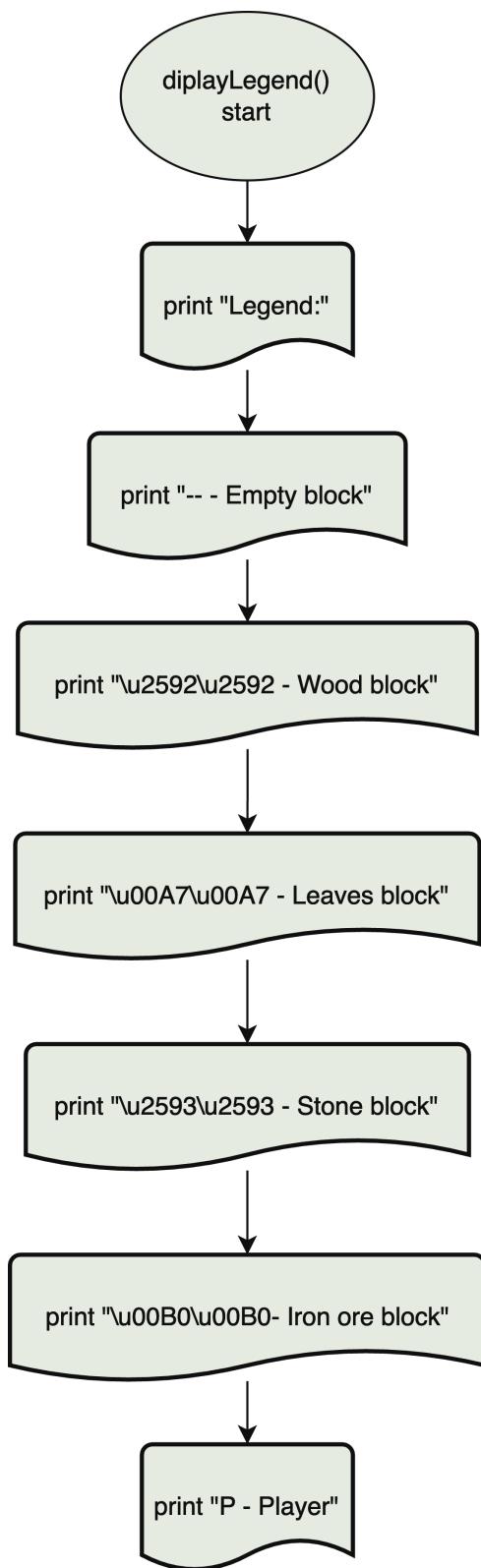
30 – getBlockName



Pseudocodes > 30-getBlockName

```
~/javacraft/Pseudocodes lockName
2      CASE block type
3          "AIR":
4              RETURN: "Empty block"
5          "WOOD":
6              RETURN: "Wood"
7          "LEAVES":
8              RETURN: "Leaves"
9          "STONE":
10         RETURN: "Stone"
11         "IRON_ORE":
12         RETURN: "Iron Ore"
13         "DEFAULT":
14         RETURN: "Unknown"
15     ENDCASE
```

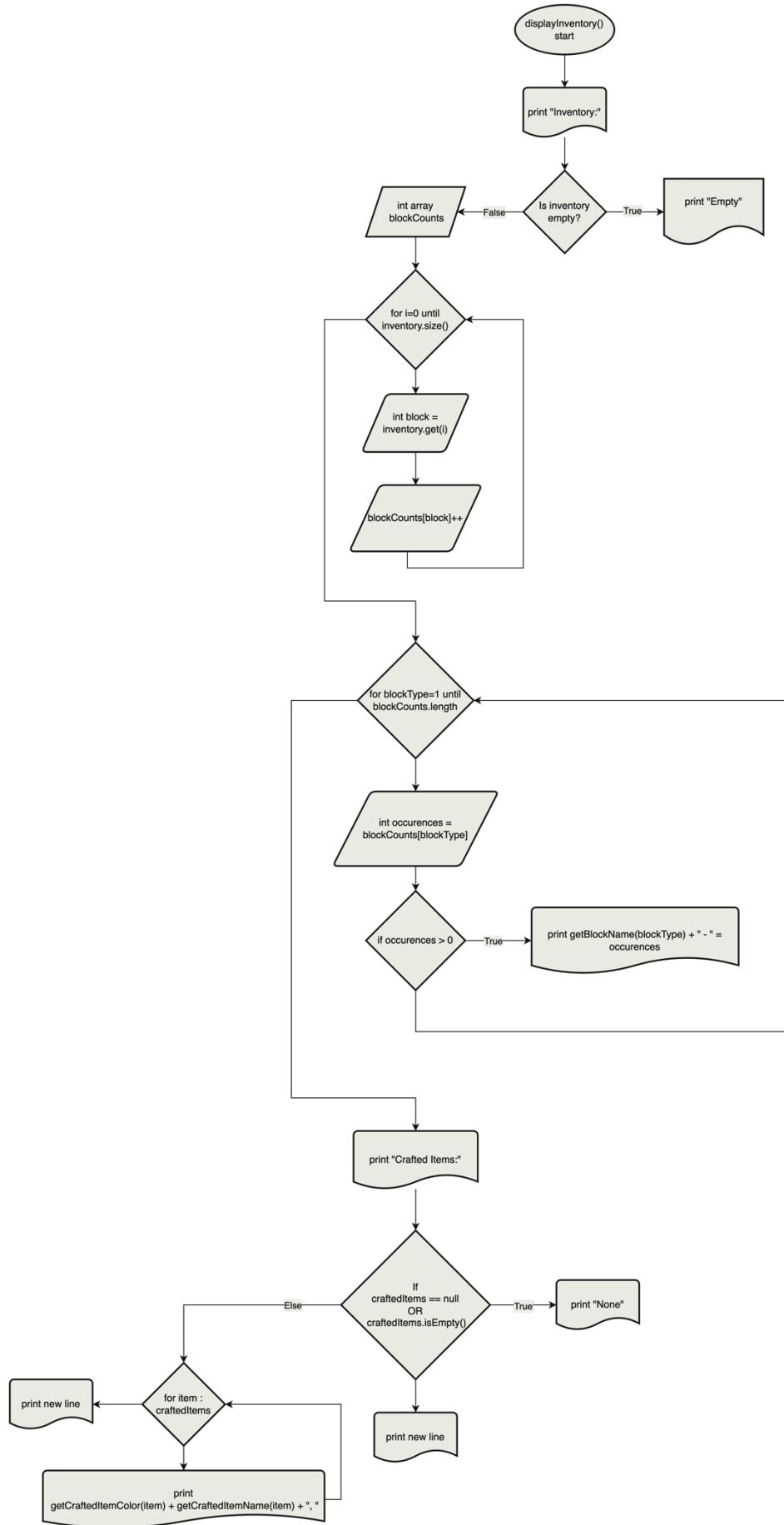
31 – displayLegend



Pseudocodes > 31-displayLegend

```
1   FUNCTION displayLegend
2       PRINT "Legend:" in blue
3       PRINT "-- - Empty block" in white
4       PRINT "█ - Wood block" in red
5       PRINT "§§ - Leaves block" in green
6       PRINT "██ - Stone block" in blue
7       PRINT "○○- Iron ore block" in white
8       PRINT "P - Player" in blue
```

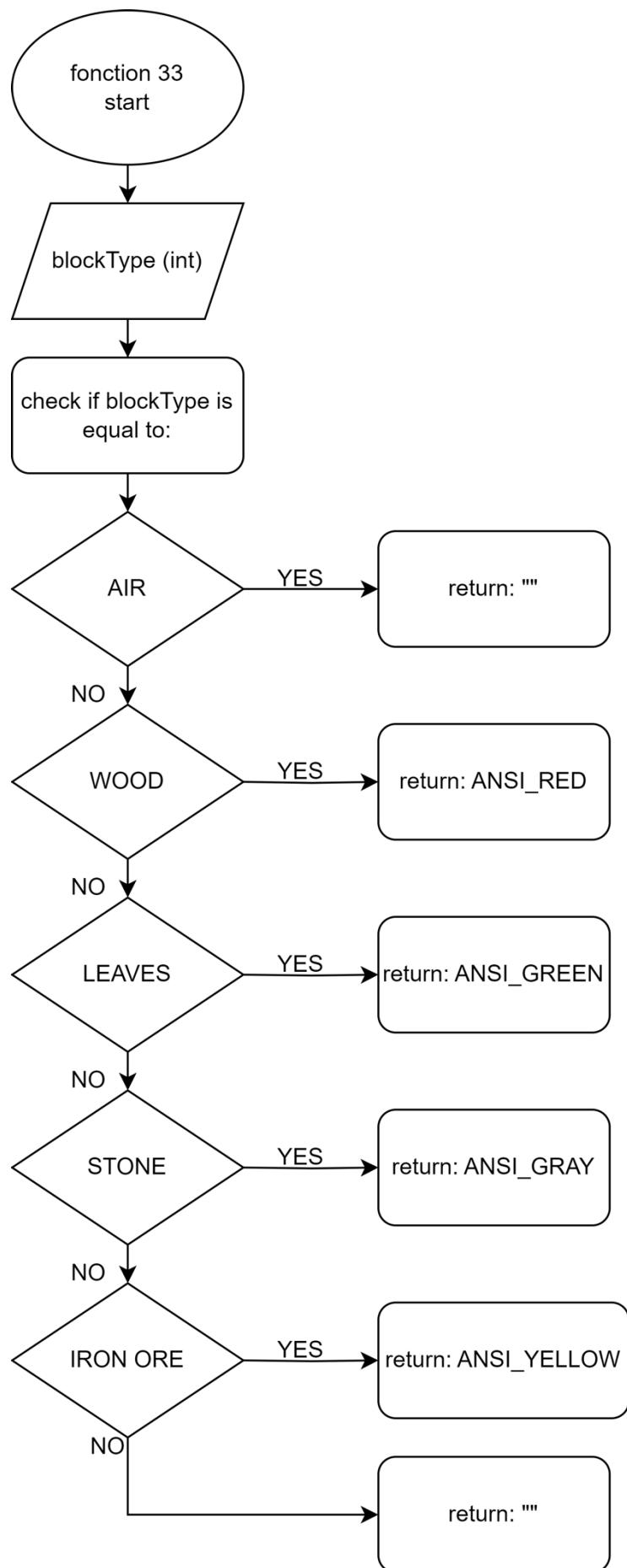
32 – displayInventory



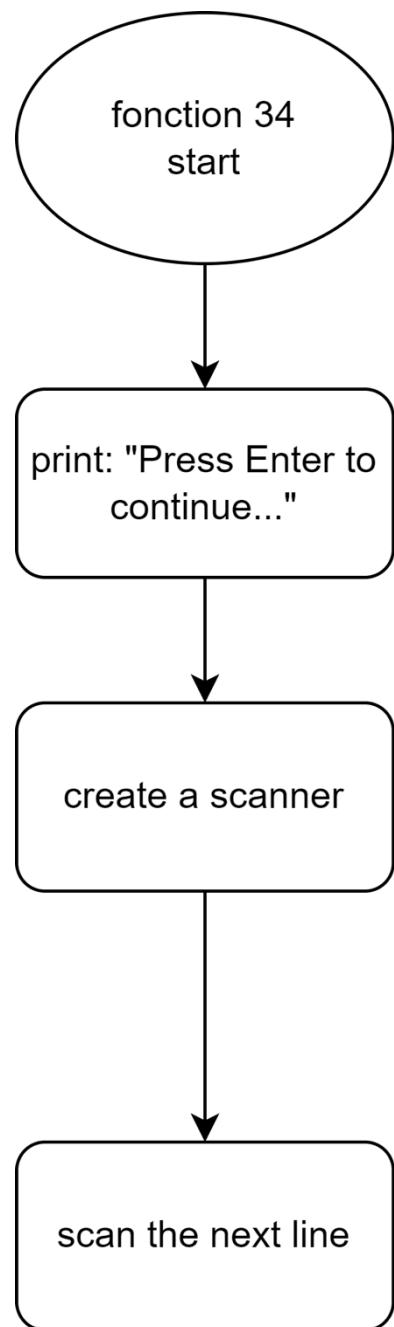
Pseudocodes > 32-displayInventory

```
1  FUNCTION displayInventory
2      PRINT "Inventory:"
3      IF inventory is empty THEN
4          PRINT "Empty" in yellow
5      ELSE
6          SET blockCounts to NEW integer array[5]
7          FOR each i in inventory
8              SET block to GET inventory(i)
9              INCREMENT blockCounts[block]
10         FOR each blockType in blockCounts
11             SET occurrences to blockCounts[blockType]
12             IF occurrences bigger than 0 THEN
13                 PRINT getBlockName(blockType) + " - " + occurrences
14             PRINT "Crafted Items:"
15             IF craftedItems equals to null THEN
16                 PRINT "None" in yellow
17             ELSE
18                 FOR each item in craftedItems
19                     PRINT getCraftedItemColor(item) + getCraftedItemName(item) + ", "
20             PRINT NEW LINE
21             PRINT NEW LINE
```

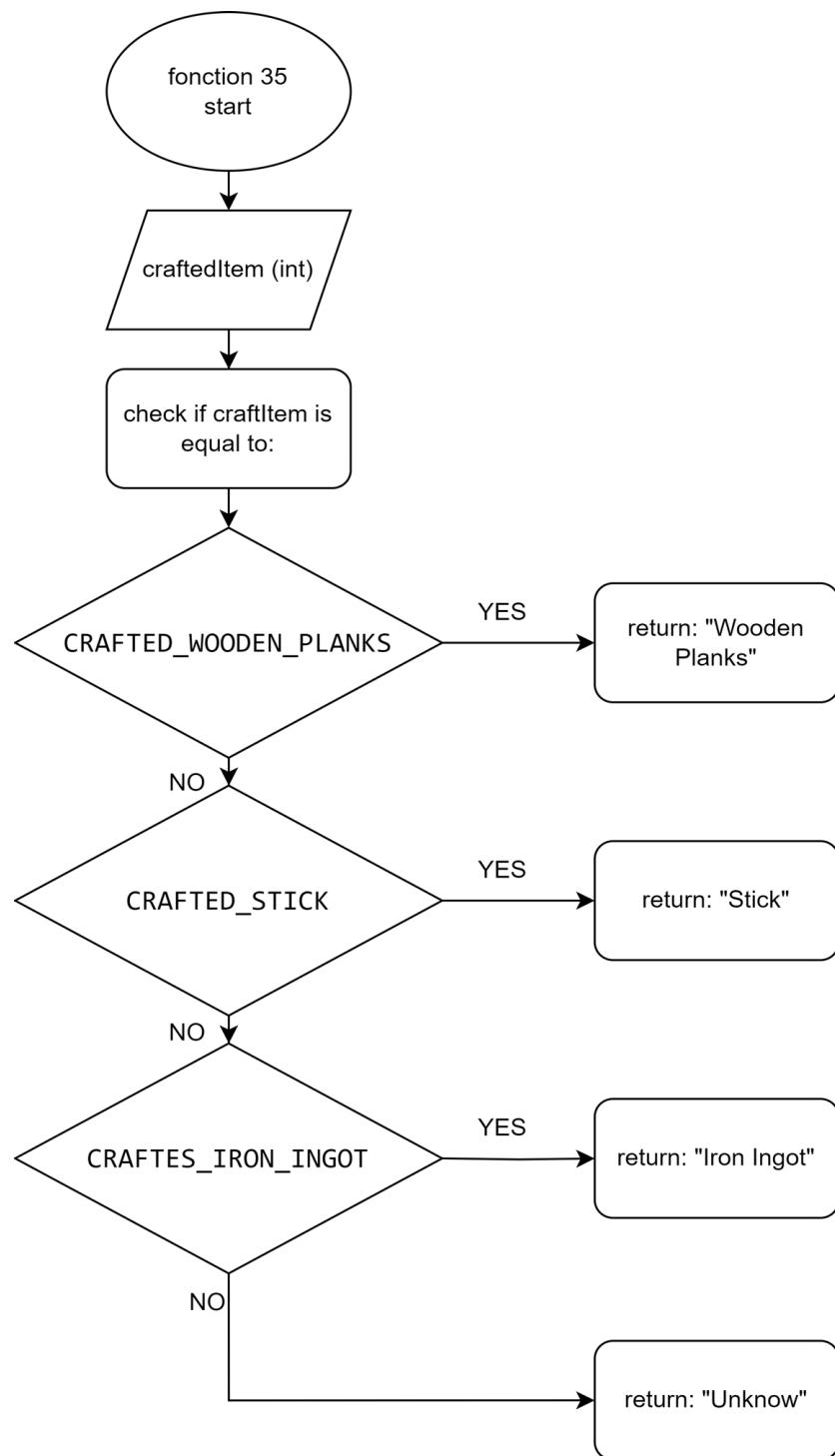
33 – getBlockColor



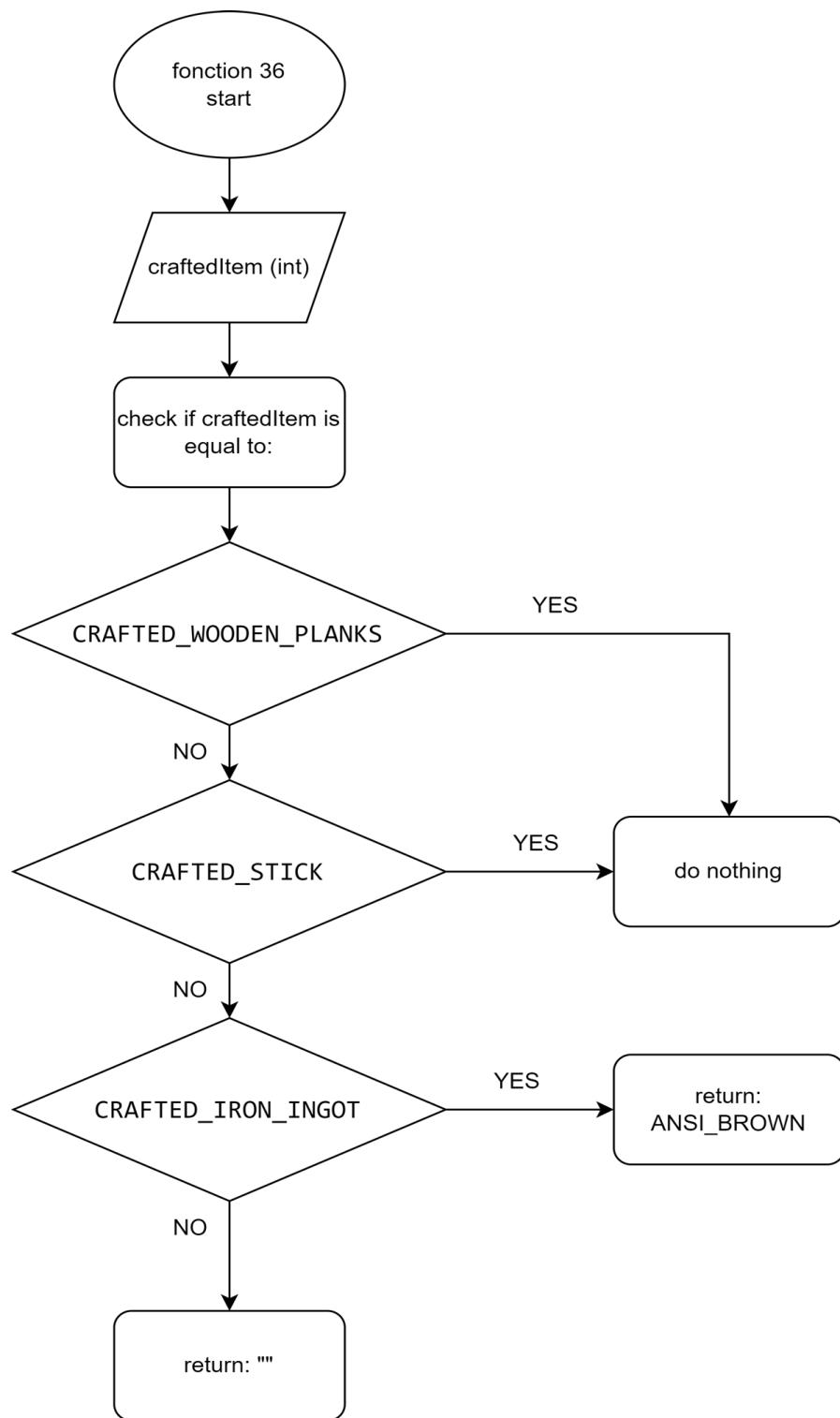
34 – waitForEnter



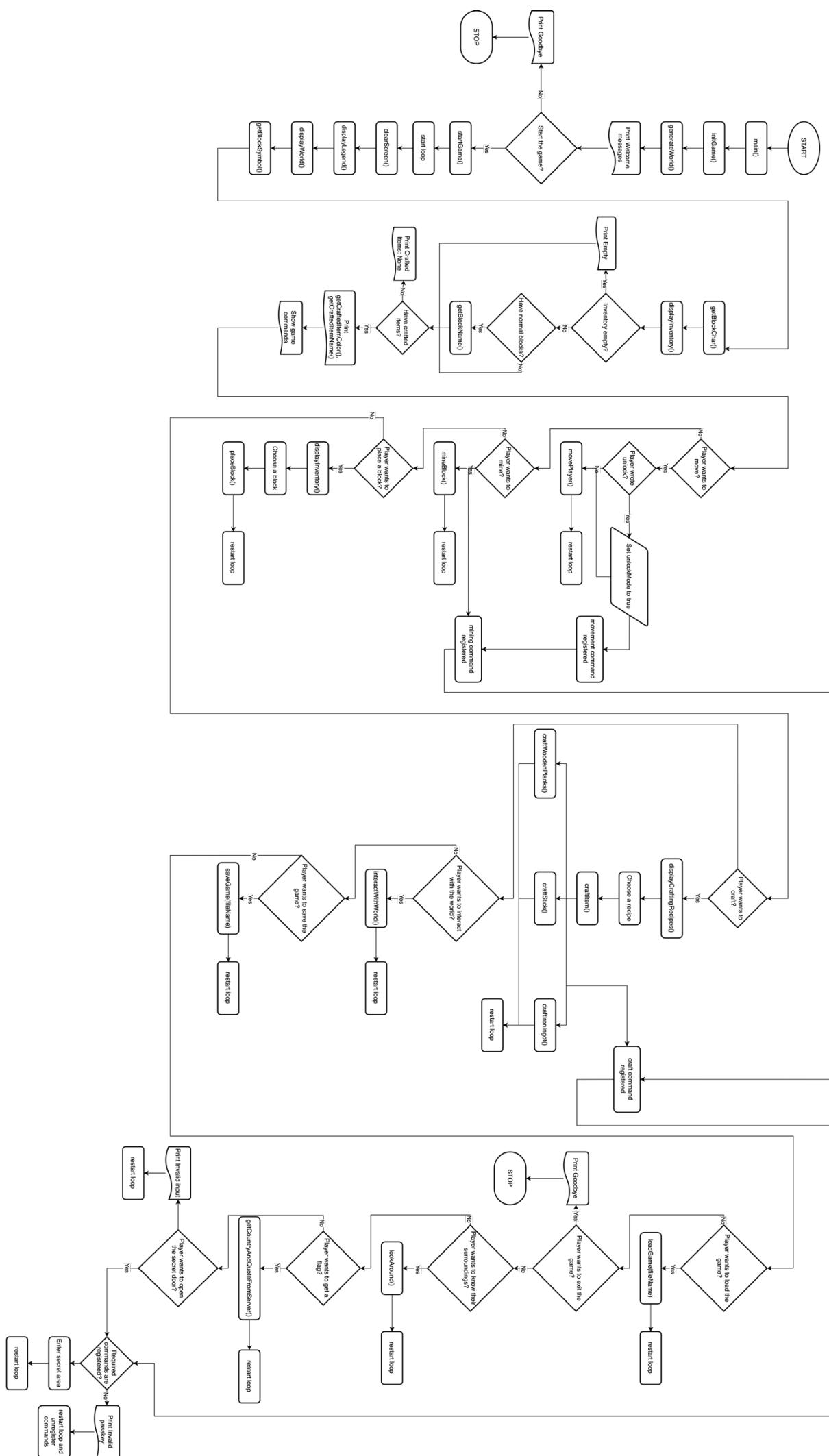
35 – getCraftedItemName



36 – getCraftedItemColor



Flowchart for the game:



Pseudocode For Game:

```
create JavaCraft class
    create main function
        call initGame function
        call generateWorld function
        print the instructions of the game with the commands
        ask to the player if he want to start the game
        if yes
            call startgame function
        else
            print: game not started. Goodbye!

create initGame function
    set the world size
    set the initial player's position
    create inventory

create generateWorld function
    create a random world

create displayWorld function
    print the world map

create getBlockSymbol function
    return the color of the block and call getBlockChar for the same block

create getBlockChar function
    return the symbol of the block

create startGame function
    call clearScreen function
    call displayLegend function
    call displayWorld function
    call displayInventory function
    print the instructions
    do what the player's action

    if the player move
        call movePlayer function
    else if the player mine
        call mineBlock function
    else if the player place a block
        ask which block he want to place
        call placeBlock function
    else if the player craft
        ask which item he want to craft
        call craftItem function
    else if the player interact with the world
        call interactWithWorld function
    else if the player save
        ask for the file name to save the game
        call saveGame function
    else if the player load
        ask which load he want to load
        call loadGame function
    else if the player exit
        print: Exiting the game. Goodbye!
        stop the game
    else if the player look
        call lookAround function
    else if the player unlock
        set unlockMode to true
    else if the player ask getflag
        call getCountryAndQuoteFromServer function
        call waitForEnter function
    else if the player open
        if all the conditions are true
            call resetWorld function
            print: Secret door unlocked!
            call waitForEnter function
        else
            print: Invalid passkey, Try again!
            call waitForEnter function
            set all the conditions to false
    else
        print: Invalid input, Please try again.

    if the first condition is true
        check if the others conditions are true too
    if secretDoorUnlocked is true
        call clearScreen function
        print the context
        set inSecretArea to true
        call resetWorld function
        set secretDoorUnlocked to false
        call fillInventory function
        call waitForEnter function

create fillInventory function
    clear the inventory
    add blockType in the inventory

create resetWorld function
    call generateEmptyWorld function
    set the player's initial position

create generateEmptyWorld function
    create a new world
    divide in tree strip the world from the top to the bot
    fill each strip with a different color, to make the dutch flag

create clearScreen function
    clear the screen'

create lookAround function
    print the context
    print the blocks next to the player
    call waitForEnter function

create movePlayer function
    move the player the way he ask

create mineBlock function
    if the block on the player's location is a air block
        Print: No block to mine here.
    if not
        add the block to his inventory
        replace the original block by an air block
```

```

        print: "Mined " and call getBlockName function with the blockType in parameter
call waitForEnter function

create placeBlock function
    check if the inventory contain the block asked
        if yes
            remove the block from the inventory
            place the block
            print: "Placed " and call getBlockName function and print: " at your position."
        else
            print that the player don't have the block by using getBlockName function
        if the block that the player want to place doesn't exist
            print: Invalid block number. Please enter a valid block number.
            print the block's numbers infos
    call waitForEnter function

create getBlockTypeFromCraftedItem function
    return the number of the block asked

create getCraftedItemFromBlockType function
    return the name of the item ask

create displayCraftingRecipes function
    print the crafting recipes

create craftItem function
    if the player choose recipes 1
        call craftWoodenPlanks function
    if the player choose recipes 2
        call craftStick function
    if the player choose recipes 3
        call craftIronIngot function
    else
        print: Invalid recipes number.
    call waitForEnter function

create craftWoodenPlanks function
    call inventoryContains function with for parameters the recipes of the item
    if the inventoryContain function return true
        call removeItemsFromInventory function with the recipes of the item
        call addCraftedItem function with for parameter the item just crafted
        Print: Crafted Wooden Planks.
    else
        print insufficient resources to craft wooden Planks.

create craftStick function
    call inventoryContains function with for parameters the recipes of the item
    if the inventoryContain function return true
        call removeItemsFromInventory function with the recipes of the item
        call addCraftedItem function with for parameter the item just crafted
        print: Crafted Stick.
    else
        print insufficient resources to craft Stick.

create craftIronIngot function
    call inventoryContains function with for parameters the recipes of the item
    if the inventoryContain function return true
        call removeItemsFromInventory function with the recipes of the item
        call addCraftedItem function with for parameter the item just crafted
        Print: Crafted Iron Ingot.
    else
        print insufficient resources to craft Iron Ingot.

create inventoryContains function
    return
        true if inventory contains items
        false if inventory does not contain items

create inventoryContains function for adding count in inventory
    itemCount is equal to 0
    for all i within inventory
        if i is equal to item
            increment itemCount
            if itemCount is equal to count
                inventoryContains is true

create removeItemsFromInventory function
    removedCount is equal to 0
    iterator is equal to inventory with the iteration function called into it
    while iterator has a token
        i is equal to iterator with next method called into it
        if i is equal to item
            remove item from inventory by iterator
            increments for removedCount
        if removedCount is equal to count
            break loop

create addCraftedItem function
    if craftedItems is empty
        create an array list for craftedItems
        add craftedItem to craftedItem

create interactWithWorld function
    switch for blockType
        case wood
            print where you obtained the wood
            add wood block to inventory
        case leaves
            print where you obtained the leaves
            add leaves block to inventory
        case stone
            print where you obtained the stone
            add stone block to inventory
        case iron ore
            print where you obtained the iron ore
            add iron ore to inventory
        case air
            print "Nothing to interact with here"
        default
            print "Unrecognized block. Cannot interact here"
    call waitForEnter function to wait for user confirmation

create saveGame function
    takes a string of the name of a file as a parameter from which to save the game
    serializes the game state data for saving
    prints where the game state was saved into
    prints whether if there is an error while saving the game
    calls waitForEnter function to ask for user confirmation

create loadGame function
    takes a string of the name of a file as a parameter to load the game from a file
    obtains the game state data from the file and loads the game
    prints the origin of the file from which the game was loaded from
    prints whether if there is an error when loading the game
    calls waitForEnter function to ask for user confirmation

create getBlockName function
    assigns to which type of block does a block belong to
    prints the name for the block

```

```

create displayLegend function
    prints out the legend for all the symbols that appear in the game

create displayInventory function
    informs the user about the contents of the inventory
    informs the user about the items that have been crafted by the user

create getBlockColor function
    assigns a color for each type of block that exists in the game

create waitForEnter function
    prints in the terminal "Press enter to continue.."
    scan the enter input of user

create getCraftedItemName function
    prints the name of the items that can be crafted into the terminal

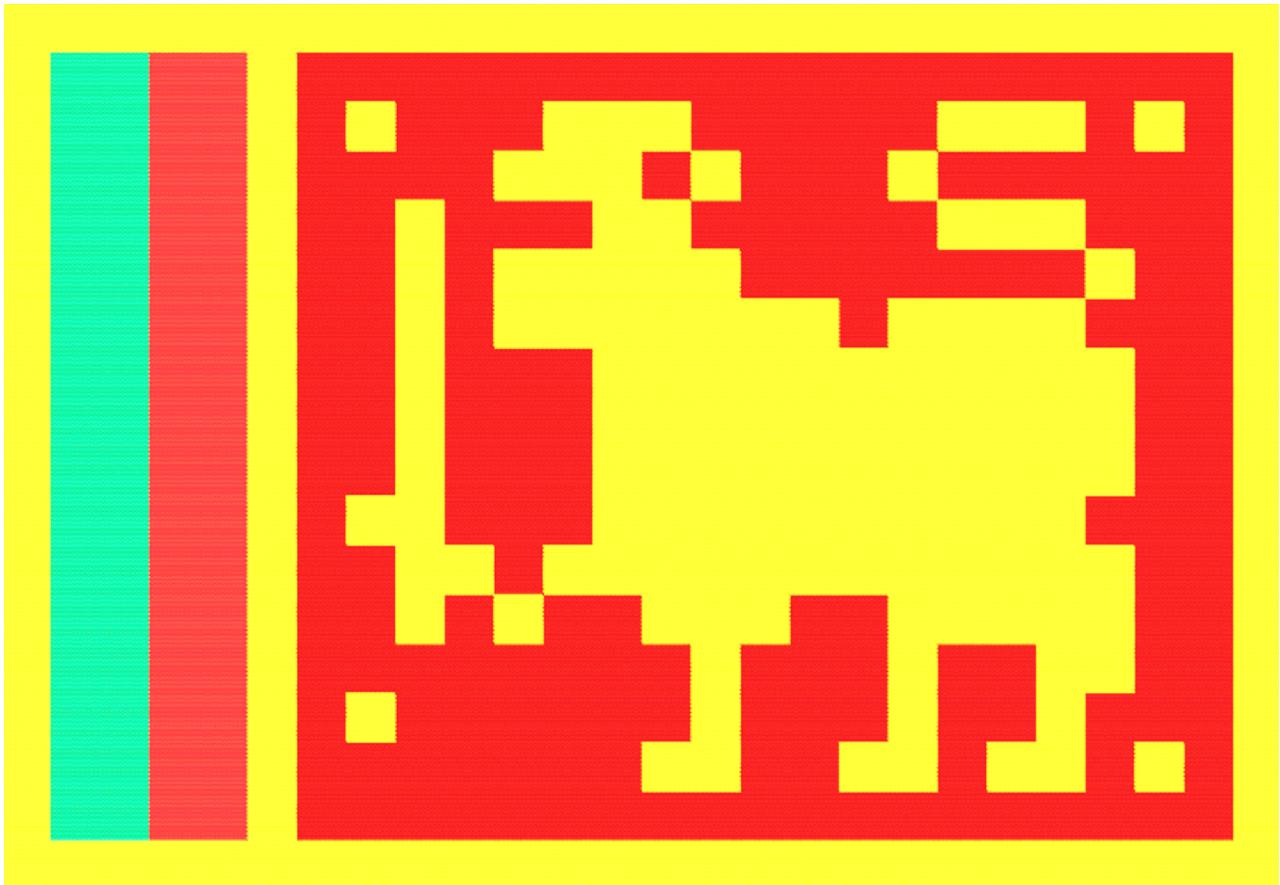
create getCraftedItemColor function
    get the color for the items that can be crafted

create getCountryAndQuoteFromServer function
    try the following for errors
        create a variable that is an URL: "https://flag.ashish.nl/get\_flag"
        create conn variable responsible for opening the connection to URL
        set the request method "POST"
        set request property with key: "Content-Type", value: "application/json"
        set do output to post requests
        string payload
        call new OutputStreamWriter on conn to convert string streams to byte streams
        write payload into the stream
        flush payload into the output stream
        close stream
        create variable reader that reads text for the connection conn to URL
        create a modifiable string called sb
        create a string called line
        while line is equal to line of text from console and it is not empty
            append sb to line
        string json is set equal to sb converted to String
        countryStart is equal to the first occurrence of " " in json plus 11
        countryEnd is equal to the first occurrence of " " starting from countryStart
        country is equal to the substring of json from position countryStart to countryEnd
        quoteStart is equal to the first occurrence of " " in json plus 9
        quoteEnd is equal to the first occurrence of " " in json starting from quoteStart
        quote is equal to the substring of json from quoteStart to quoteEnd
        replace " " in quote with "
        print " and country
        print " and quote

if there is an error
    print where in the code the error occurred
    Print "Error connecting to the server"

```

The flag:



Code snippets:

- We modified the interactWithWorld() function so it would check for the cow:

```
722     public static void interactWithWorld() {  
723         int blockType = world[playerX][playerY];  
724         if (cowX == playerY && cowY == playerX) {  
725             System.out.println(x:"You kill the cow and get its meat (sad cow noises > increase the volume)");  
726             inventory.add(MEAT);  
727  
728             String filePath =  
729             "cow.wav";  
730             Player play = new Player();  
731             play.playMusic(filePath);  
732  
733             waitForEnter();  
734             return;  
735         }  
    }
```

- We added functions to deal with the meat interactions:

```
659     public static void cookedMeat() {  
660         if (furnaceOwned != true) {  
661             System.out.println(x:"You need a Furnace to cook the meat.");  
662         } else {  
663             if (inventoryContains(MEAT)) {  
664                 removeItemsFromInventory(MEAT, count:1);  
665                 addCraftedItem(COOKED_MEAT);  
666                 System.out.println(x:"Meat is cooked.");  
667             } else {  
668                 System.out.println(x:"Insufficient resources to craft Crafting Table.");  
669             }  
670         }  
671     }  
672  
672     public static void eatCowMeat(){  
673         if (inventoryContains(COOKED_MEAT)) {  
674             removeItemsFromInventory(COOKED_MEAT, count:1);  
675             System.out.println(x:"You have eaten meat!");  
676             waitForEnter();  
677         } else {  
678             System.out.println(x:"Insufficient resources to Eat Meat");  
679             waitForEnter();  
680         }  
681     }  
}
```

- We extended the list of the craftable items and added some new functions to create them:

```

627     public static void craftCraftingTable() {
628         if (craftedItems.contains(CRAFTED_WOODEN_PLANKS)) {
629             for (int i=0; i <=4; i++) {
630                 craftedItems.remove(Integer.valueOf(CRAFTED_WOODEN_PLANKS));
631             }
632             addCraftedItem(CRAFTED_CRAFTING_TABLE);
633             craftingTableOwned = true;
634             System.out.println("Crafted Crafting Table.");
635         } else {
636             System.out.println("Insufficient resources to craft Crafting Table.");
637         }
638     }
639
640     public static void craftFurnace() {
641         if (craftingTableOwned != true) {
642             System.out.println("You need a Crafting Table to craft a Furnace");
643         } else {
644             if (inventoryContains(COAL, count:2)) {
645                 if (inventoryContains(IRON_ORE)) {
646                     removeItemsFromInventory(COAL, count:2);
647                     removeItemsFromInventory(IRON_ORE, count:1);
648                     addCraftedItem(CRAFTED_FURNACE);
649                     furnaceOwned = true;
650                     System.out.println("Crafted Furnace.");
651                 }
652             } else {
653                 System.out.println("Insufficient resources to craft Furnace.");
654             }
655         }
656     }
657 }
```

- We updated the initGame() function so it would “spawn” the cow in a random location:

```

104     public static void initGame(int worldWidth, int worldHeight) {
105         JavaCraft.worldWidth = worldWidth;
106         JavaCraft.worldHeight = worldHeight;
107         JavaCraft.world = new int[worldWidth][worldHeight];
108         playerX = worldWidth / 2;
109         playerY = worldHeight / 2;
110         int minXCow = 0, maxXCow = worldWidth, minYCow = 0, maxYCow = worldHeight;
111         cowX = (int) Math.floor(Math.random() * (maxXCow - minXCow + 1) + minXCow);
112         cowY = (int) Math.floor(Math.random() * (maxYCOW - minYCow + 1) + minYCow);
113         inventory = new ArrayList<>();
114     }
115 }
```

- We modified the displayWord() function so it would display the cow:

```

140     public static void displayWorld() {
141         System.out.println(ANSI_CYAN + "World Map:" + ANSI_RESET);
142         System.out.println("═" + "=" .repeat(worldWidth * 2 - 2) + "═");
143         for (int y = 0; y < worldHeight; y++) {
144             System.out.print("║");
145             for (int x = 0; x < worldWidth; x++) {
146                 if (x == playerX && y == playerY && !inSecretArea) {
147                     System.out.print(ANSI_GREEN + "P " + ANSI_RESET);
148                 } else if (x == playerX && y == playerY && inSecretArea) {
149                     System.out.print(ANSI_BLUE + "P " + ANSI_RESET);
150                 } else if (x == cowY && y == cowX && !inSecretArea) {
151                     System.out.print(ANSI_PURPLE + "C " + ANSI_RESET);
152                 } else {
153                     System.out.print(getBlockSymbol(world[x][y]));
154                 }
155                 System.out.print(ANSI_RESET);
156             }
157         }
158         System.out.println("═" + "=" .repeat(worldWidth * 2 - 2) + "═");
159     }
160 }
161 }
```

- We added a function to move the cow on every 3rd interaction in a random direction:

```

163     public static void moveCow() {
164         // Move the cow in accordance to a random generated number --> use Math.random()
165         int min = 0, max = 3;
166         int movementCow = (int) Math.floor(Math.random() * (max - min + 1) + min);
167
168         // 0 --> Up, 1 --> down, 2 --> left, 3 --> right
169
170         switch (movementCow) {
171             case 0:
172                 if (cowY > 0) {
173                     cowY--;
174                 }
175                 break;
176             case 1:
177                 if (cowY < worldHeight - 1) {
178                     cowY++;
179                 }
180                 break;
181             case 2:
182                 if (cowX > 0) {
183                     cowX--;
184                 }
185                 break;
186             case 3:
187                 if (cowX < worldWidth - 1) {
188                     cowX++;
189                 }
190                 break;
191             default:
192                 break;
193         }
194     }
195 }
```