Project JavaCraft

Project Report: 70

Friday, October 20, 2023

Table of contents

Attribute	Details
Group Name	Group70
Group Number	70
TA	[Name]

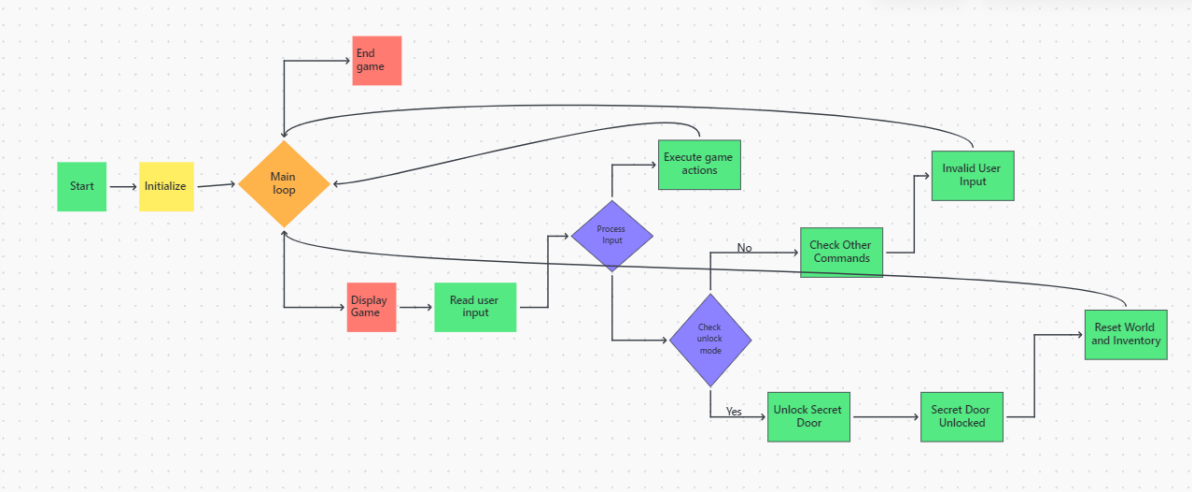
Student Name	Student ID
[Amir]	[i6356969]
[Louis]	[i6365585]
[Markos]	[i6354900]
[Stefania]	[i6347009]

1 Introduction

[Provide a brief introduction to the report here.]

2 JavaCraft's Workflow

- Flowchart For Game:



- Pseudocode For Game:

Function **main**:

- Call initGame with parameters 25 and 15
- Call generateWorld
- Print "Welcome to Simple Minecraft!"
- Print "Instructions:"
- Print " - Use 'W', 'A', 'S', 'D', or arrow keys to move the player."
- Print " - Press 'P' to place a block from your inventory at your position."
- Print " - Press 'C' to view crafting recipes and 'I' to interact with elements in the world."
- Print " - Press 'Save' to save the game state and 'Load' to load a saved game state."
- Print " - Press 'Exit' to quit the game."
- Print " - Type 'Help' to print these instructions again."
- Create a Scanner object named 'scanner' for user input
- Print "Start the game? (Y/N): "
- Read user input into a variable 'startGameChoice' and convert it to uppercase
- If 'startGameChoice' equals "Y":
- Call startGame

- Else:

  Print "Game not started. Goodbye!"

Function **initGame** with parameters '**worldWidth**' and '**worldHeight**':

- Set 'JavaCraft.worldWidth' to 'worldWidth'
- Set 'JavaCraft.worldHeight' to 'worldHeight'
- Create a new 2D array 'JavaCraft.world' with dimensions 'worldWidth' x 'worldHeight'
- Set 'JavaCraft.playerX' to 'worldWidth / 2'
- Set 'JavaCraft.playerY' to 'worldHeight / 2'
- Initialize 'JavaCraft.inventory' as an empty list

Function **generateWorld**:

- Create a new Random object named 'rand'
- Loop 'y' from 0 to 'JavaCraft.worldHeight':
- Loop 'x' from 0 to 'JavaCraft.worldWidth':
- Generate a random integer 'randValue' between 0 and 99 using 'rand.nextInt(100)'
- If 'randValue' is less than 20:

  Set 'JavaCraft.world[x][y]' to 'WOOD'

- Else if 'randValue' is less than 30:

  Set 'JavaCraft.world[x][y]' to 'LEAVES'

- Else if 'randValue' is less than 40:

  Set 'JavaCraft.world[x][y]' to 'STONE'

- Else if 'randValue' is less than 60:

  Set 'JavaCraft.world[x][y]' to 'IRON_ORE'

- Else if 'randValue' is less than 70:

  Set 'JavaCraft.world[x][y]' to 'GOLD_ORE'

- Else if 'randValue' is less than 80:

Set 'JavaCraft.world[x][y]' to 'DIAMOND_ORE'

- Else:

  Set 'JavaCraft.world[x][y]' to 'AIR'

Function **PrintWorld**:

- Print "World Map:"
- Print "╔═══" + "═".repeat(worldWidth * 2 - 2) + "╗"
- For 'y' from 0 to 'worldHeight':

  Print "║"

- For 'x' from 0 to 'worldWidth':

  If 'x' equals 'playerX' and 'y' equals 'playerY':

  If 'not inSecretArea':

  Print "P "

- Else:

  Print "P "

- Else:

  Print getBlockSymbol(world[x][y])

  Print "║"

  Print "╚═══" + "═".repeat(worldWidth * 2 - 2) + "╝"

3 Functionality Exploration

No.	Function Name:	Description:
1	initGame	Initializing Game Parameters
2	generateWorld	Generates the world map
3	displayWorld	Shows the user the world generation
4	getBlockSymbol	For every block type it returns a specific color and symbol
5	getBlockChar	Returns the character assigned to the block
6	startGame	Starts the Game
7	fillInventory	Fills the inventory with max amount of values
8	resetWorld	Resets the world
10	lookAround	Shows the blocks around the player
11	movePlayer	Give the ability for the player to move around
12	mineBlock	Give the ability for the player to mine a block
13	placeBlock	Give the ability for the player to place a block
14	getBlockTypeFromCraftedItem	Returns the block type of the crafted item
15	getCraftedItemFromBlockType	Returns crafted item's type
16	displayCraftingRecipes	Displays crafting recipe

17	craftItem	Crafts item by using materials
18	craftWoodenPlanks	Crafts wooden planks out of the wood blocks in the inventory
19	craftStick	Crafts Stick if possible
20	craftIronIngot	Crafts iron ingot out of iron ores from the inventory
21	inventoryContains	Shows the items in the inventory
22	removeItemsFromInventory	Removes specific items after crafting another item
23	addCraftedItem	Adds a crafted item to the inventory
24	interactWithWorld	Allows the user to interact with objects
25	saveGame	Saves the current game session
26	loadGame	loads a saved game
27	getBlockName	Returns the block name
28	displayLegend	Displays legendre;
29	displayInventory	Displays player's inventory
30	getBlockColor	Returns Block Color
31	waitForEnter	Command thats makes the player press enter to continue
32	getCraftedItemName	Returns crafted item's name
33	getCraftedItemColor	Returns crafted item's color
34	getCountryAndQuoteFromServer	Sends commands to the server for getting flag details using APIs

35	craftedItemsContain	Checks to see if there are certain amount of crafted items are available
36	main	Initializes the game, prints prompts and checks if game should start or not
37	removeItemsFromCrafted Items	Removes items from the crafted items list
38	craftIronPickaxe	Crafts iron pickaxe

Note:

Provide flowchart and pseudocode for at least 15 functions in the Appendix.

4 Finite State Automata (FSA) Design

- Secret Door Logic Analysis:

The secret door opens when some conditions in the game become **true**. The conditions are mentioned below:

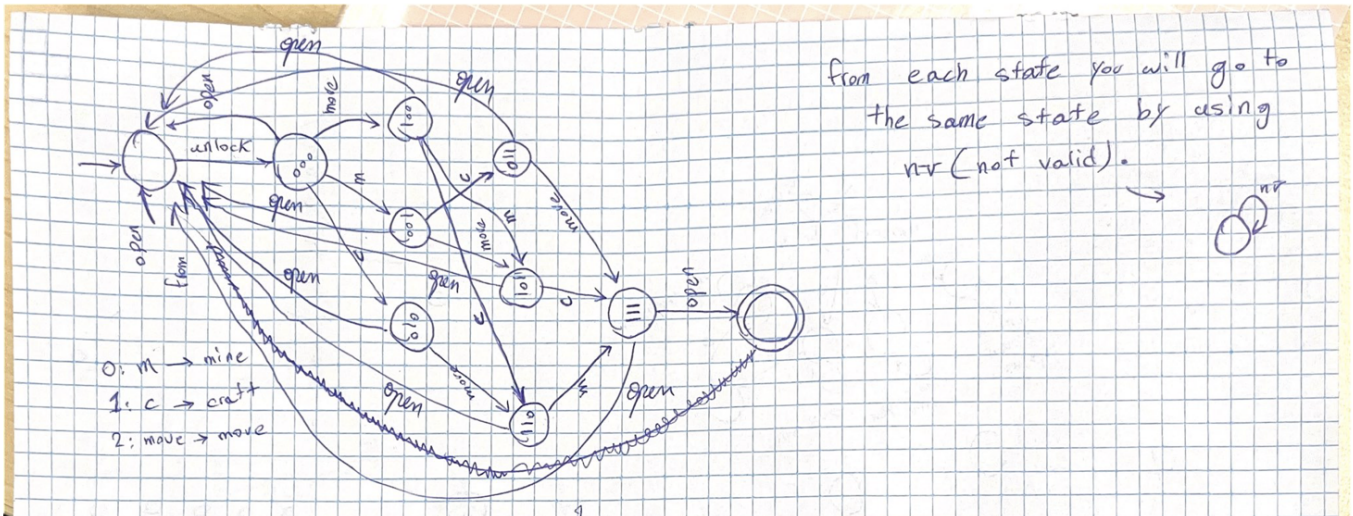
Craft - Mine - Move - Unlock - Open

First of all we need to use the command UNLOCK to begin the process. After that, we have 3 conditions that we need to make true. Each could be either 0 or 1, so there are 8(2^3) states.

Navigating between the states is demonstrated below. Keep in mind that if we use a non-acceptable command we take a step using the edge with ‘nv’(Not Valid) which means you have to enter a valid response in order to move.

When we reach the state 111, all the states are true and with the command “open” we can reach the final state(Accepting state). However, if we type something unacceptable, we go back to the start(this is true for other states as well).

- FSA Illustration & Description:



5 Git Collaboration & Version Control

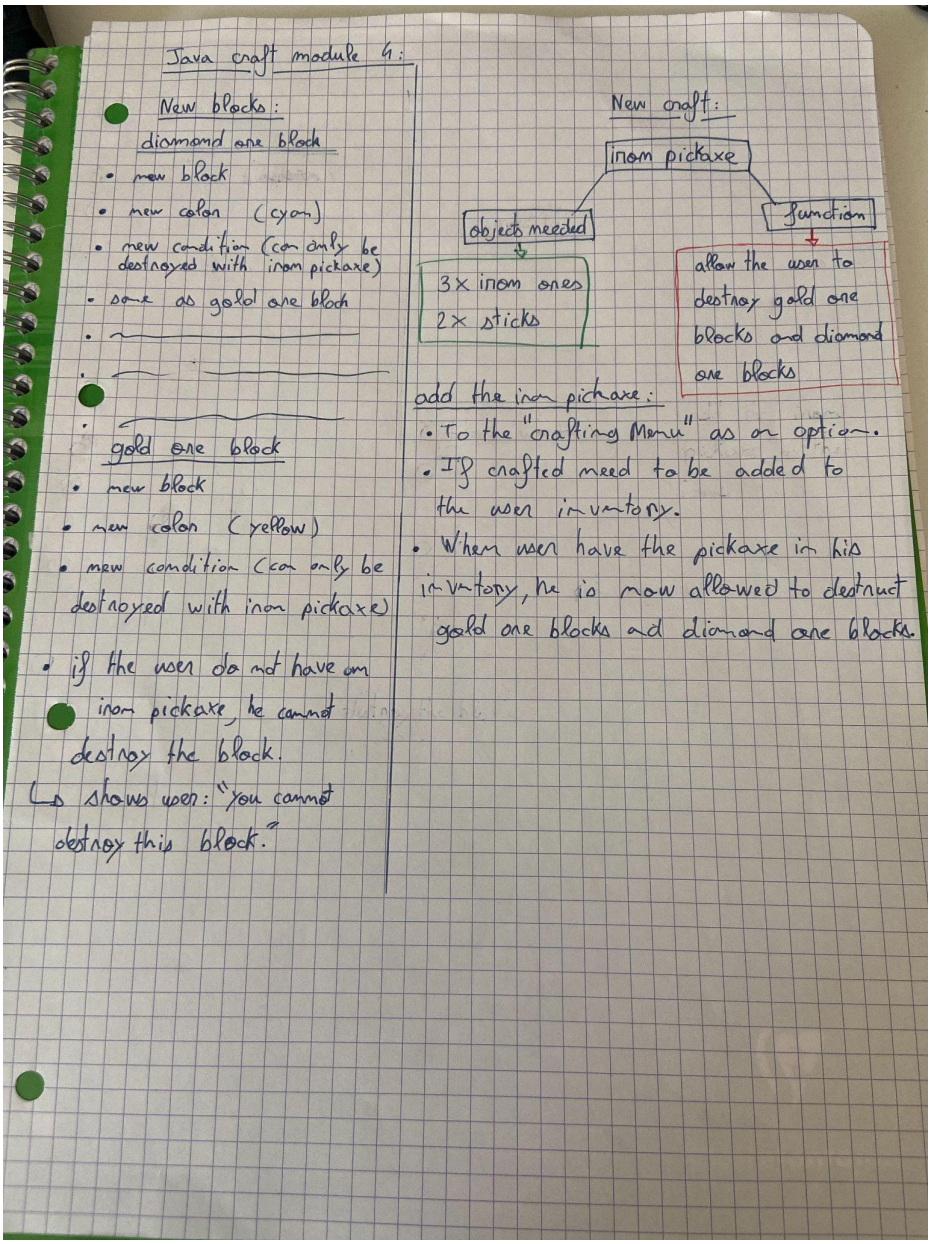
- Repository Link: Group70
- Branch Details: List branch names and corresponding members • Changes & Conflicts: Discuss how changes and conflicts were handled.

- We added a branch for our group, We tried to use one or two laptops for coding different tasks, for the final submission list of different commits could be added.

6 Extending the Game Code (For Final Submission)

We added 3 new items. Diamond Ore, Gold Ore, and Iron Pickaxe. In order to craft Diamonds or Gold, we first need to have an iron pickaxe in our inventory. In addition, with the use of Gold and Diamonds, we can craft Gold and Diamond Ingots. In order to add these items, we had to change some of the functions and add some of our own functions.

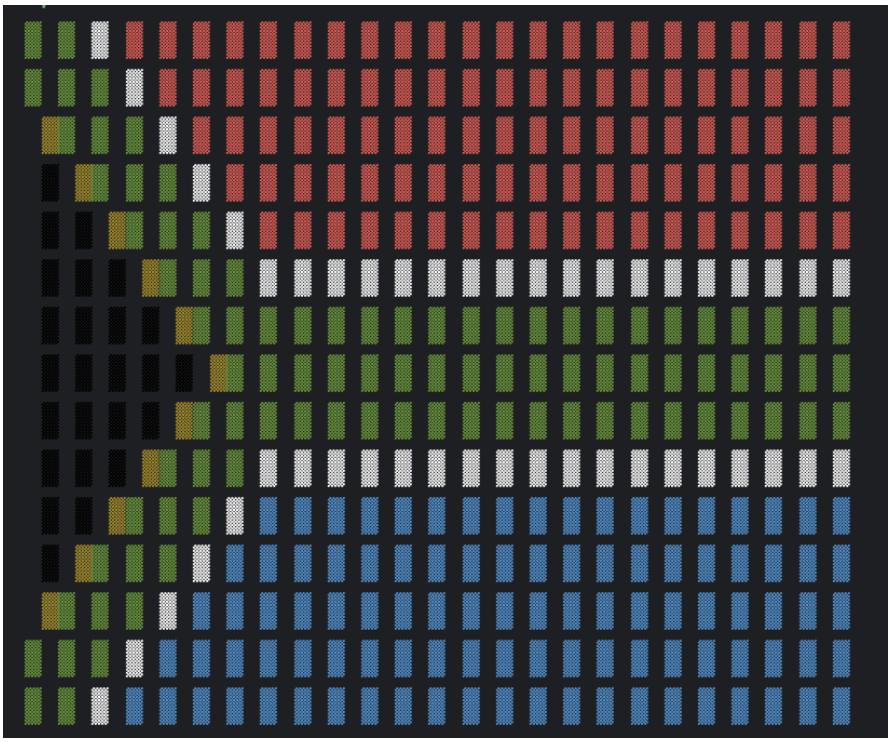
We also added functions for adding and removing items from the crafted items in order to make crafting and checking the items that we crafted easier.



7 Interacting with Flags API (For Final Submission)

We had to add the South African flag to the game instead of the NL flag. We tried to use Strings of block-shaped characters to show each pixel and used ASCII colors to change the color of each pixel.

This is a smaller version of how our flag looks like. (The final versions size is 30 by 50)



8 Conclusion (For Final Submission)

In this project, first we tried to understand what was already coded and how each function worked and interacted with the game, and then we tried to figure out the secrets and try to model that with a FSA. We tried to write pseudo-codes for different functions and illustrate their functionality using flowcharts. In order to change the code, we used git to keep track of our modifications. Lastly, we added 2 new blocks and 3 new items and some added functionalities that were not in the original game like conditions for mining gold and diamond ore and we also used APIs to send and receive data from the server in order to get our flag.

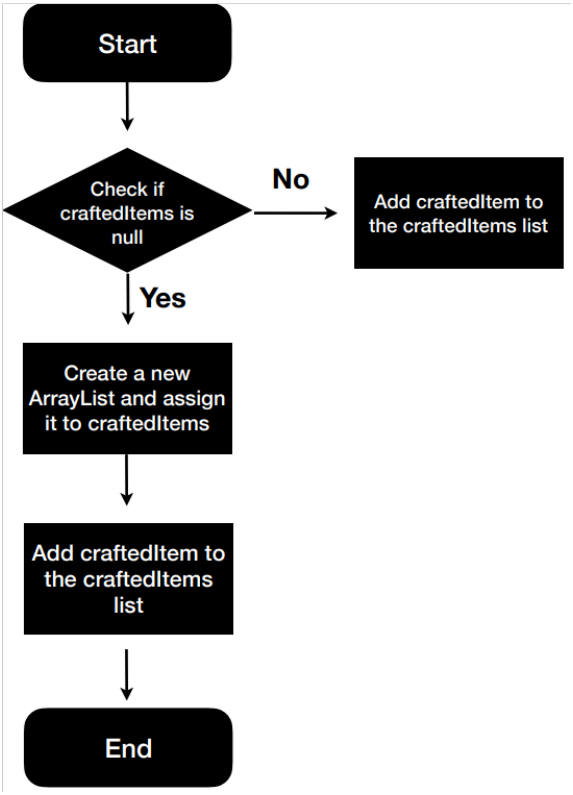
In general, we learned how to code different functions in Java and build upon existing code. Work as a team and divide different tasks, draw FSA and flowcharts, and write pseudo codes. We are also familiar with git and how to use it and know how to send HTTP requests to a server using APIs.

9 Appendix

Include any additional pseudocode, flowcharts, or supplementary material.

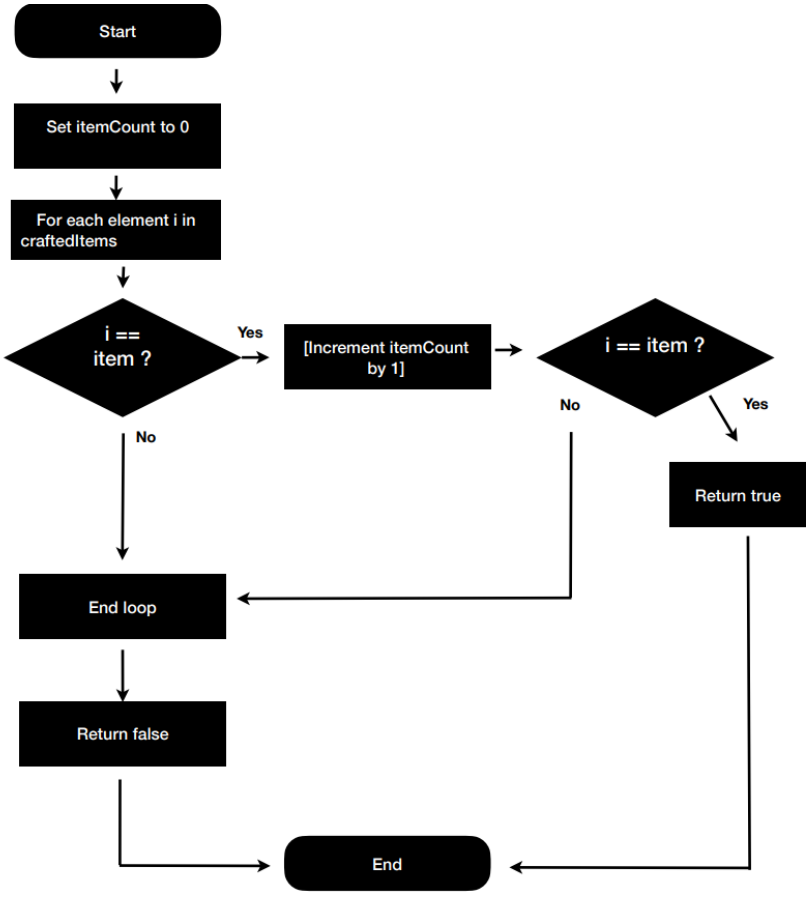
Pseudocode:

```
FUNCTION addCraftedItem(craftedItem):
   IF craftedItems IS NULL THEN
      CREATE craftedItems AS NEW ArrayList
   END IF
   ADD craftedItem TO craftedItems
END FUNCTION
```



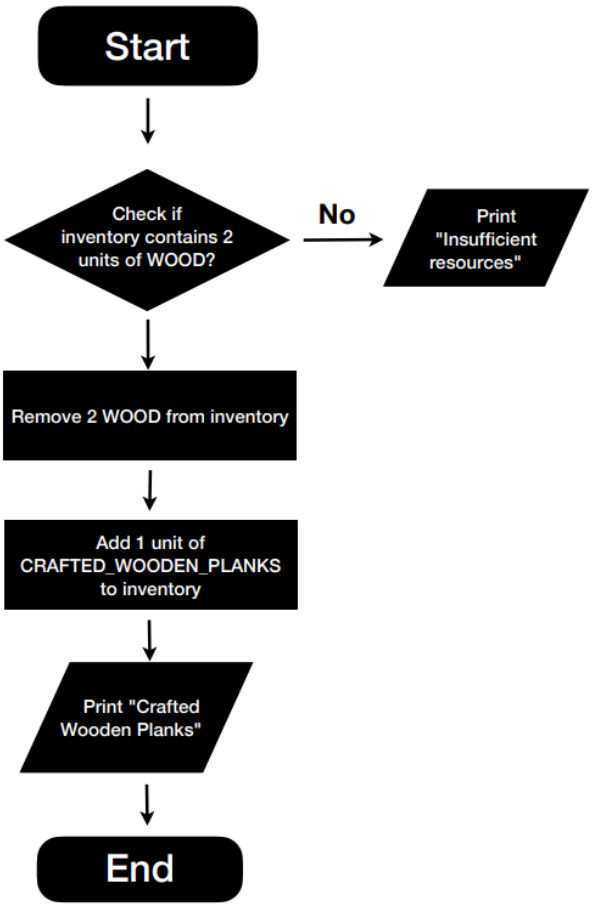
Pseudocode:

```
FUNCTION craftedItemsContains(item, count):
   SET itemCount TO 0
   FOR EACH i IN craftedItems DO
      IF i EQUALS item THEN
         INCREMENT itemCount BY 1
         IF itemCount EQUALS count THEN
            RETURN TRUE
         END IF
      END IF
   END FOR
   RETURN FALSE
END FUNCTION
```



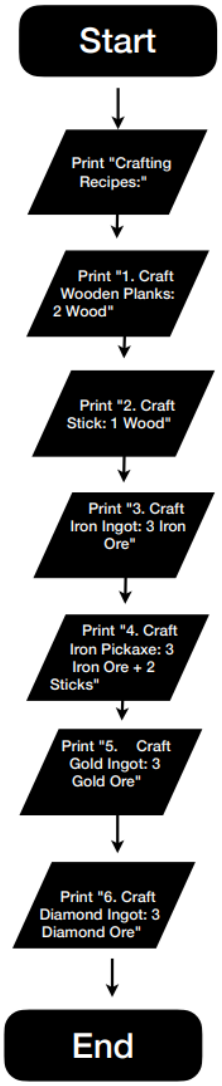
Pseudocode:

```
FUNCTION craftWoodenPlanks():
   IF inventoryContains(WOOD, 2) THEN
      CALL removeItemsFromInventory(WOOD, 2)
      CALL addCraftedItem(CRAFTED_WOODEN_PLANKS)
      PRINT "Crafted Wooden Planks."
   ELSE
      PRINT "Insufficient resources to craft Wooden Planks."
   END IF
END FUNCTION
```



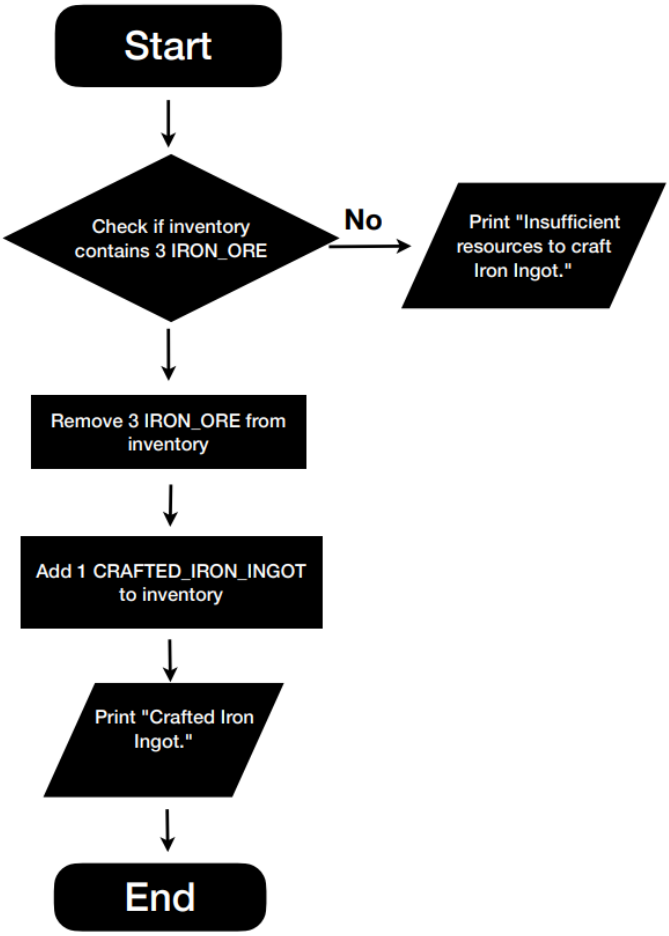
Pseudocode:

```
FUNCTION displayCraftingRecipes():
    PRINT "Crafting Recipes:"
    PRINT "1. Craft Wooden Planks: 2 Wood"
    PRINT "2. Craft Stick: 1 Wood"
    PRINT "3. Craft Iron Ingot: 3 Iron Ore"
    PRINT "4. Craft Iron Pickaxe: 3 Iron Ore + 2 Sticks"
    PRINT "5. Craft Gold Ingot: 3 Gold Ore"
    PRINT "6. Craft Diamond Ingot: 3 Diamond Ore"
END FUNCTION
```



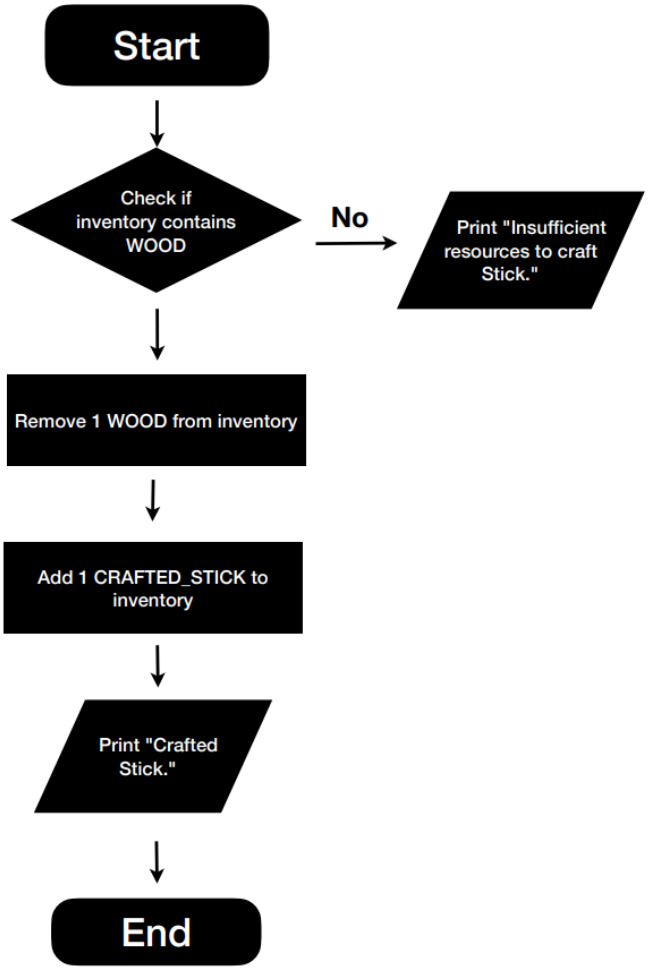
Pseudocode:

```
FUNCTION craftIronIngot():
   IF inventoryContains(IRON_ORE, 3) THEN
      CALL removeItemsFromInventory(IRON_ORE, 3)
      CALL addCraftedItem(CRAFTED_IRON_INGOT)
      PRINT "Crafted Iron Ingot."
   ELSE
      PRINT "Not enough iron ore to craft an Iron Ingot."
   END IF
END FUNCTION
```



Pseudocode:

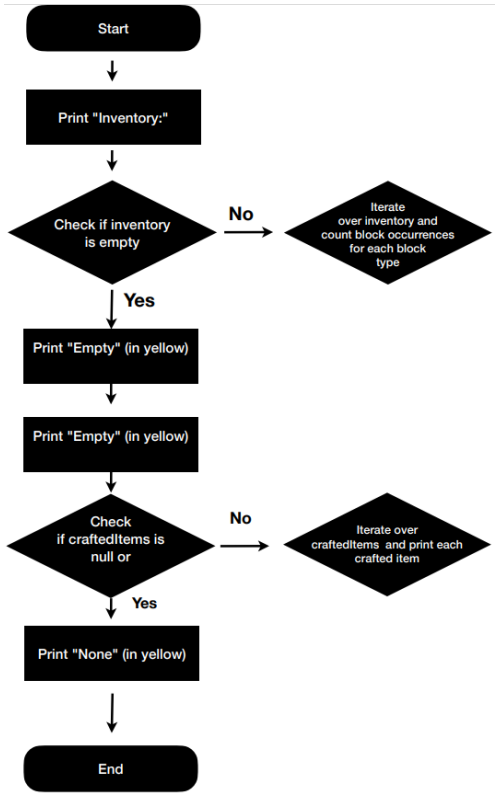
```
FUNCTION craftStick():
   IF inventoryContains(WOOD) THEN
      CALL removeItemsFromInventory(WOOD, 1)
      CALL addCraftedItem(CRAFTED_STICK)
      PRINT "Crafted Stick."
   ELSE
      PRINT "Not enough materials to make a Stick."
   END IF
END FUNCTION
```



Pseudocode:

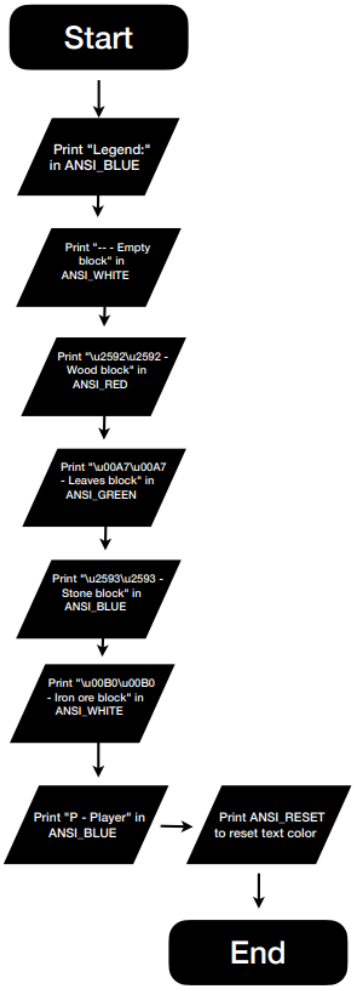
```
FUNCTION displayInventory():
   PRINT "Inventory:"
   IF inventory IS EMPTY THEN
      PRINT "Empty" IN ANSI_YELLOW
   ELSE
      DECLARE blockCounts AS ARRAY OF INTEGER WITH SIZE 10
      FOR EACH block IN inventory DO
         INCREMENT blockCounts[block] BY 1
      END FOR
      FOR blockType FROM 1 TO LENGTH(blockCounts) - 1 DO
         DECLARE occurrences AS INTEGER
         SET occurrences TO blockCounts[blockType]
         IF occurrences > 0 THEN
            PRINT getBlockName(blockType) + " - " + occurrences
         END IF
      END FOR
   END IF

   PRINT "Crafted Items:"
   IF craftedItems IS NULL OR craftedItems IS EMPTY THEN
      PRINT "None" IN ANSI_YELLOW
   ELSE
      FOR EACH item IN craftedItems DO
         PRINT getCraftedItemColor(item) + getCraftedItemName(item) +
", " + ANSI_RESET
      END FOR
   END IF
END FUNCTION
```



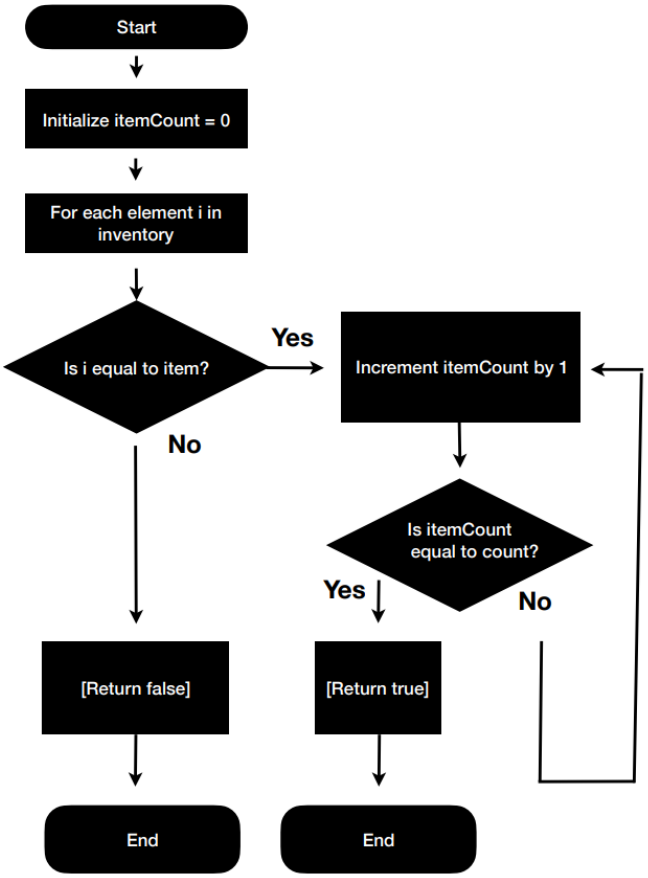
Pseudocode:

```
FUNCTION displayLegend():
    PRINT "Legend:"
    PRINT "-- - Empty block"
    PRINT "^^ - Wood block"
    PRINT "## - Leaves block"
    PRINT "## - Stone block"
    PRINT "oo - Iron ore block"
    PRINT "P - Player"
END FUNCTION
```



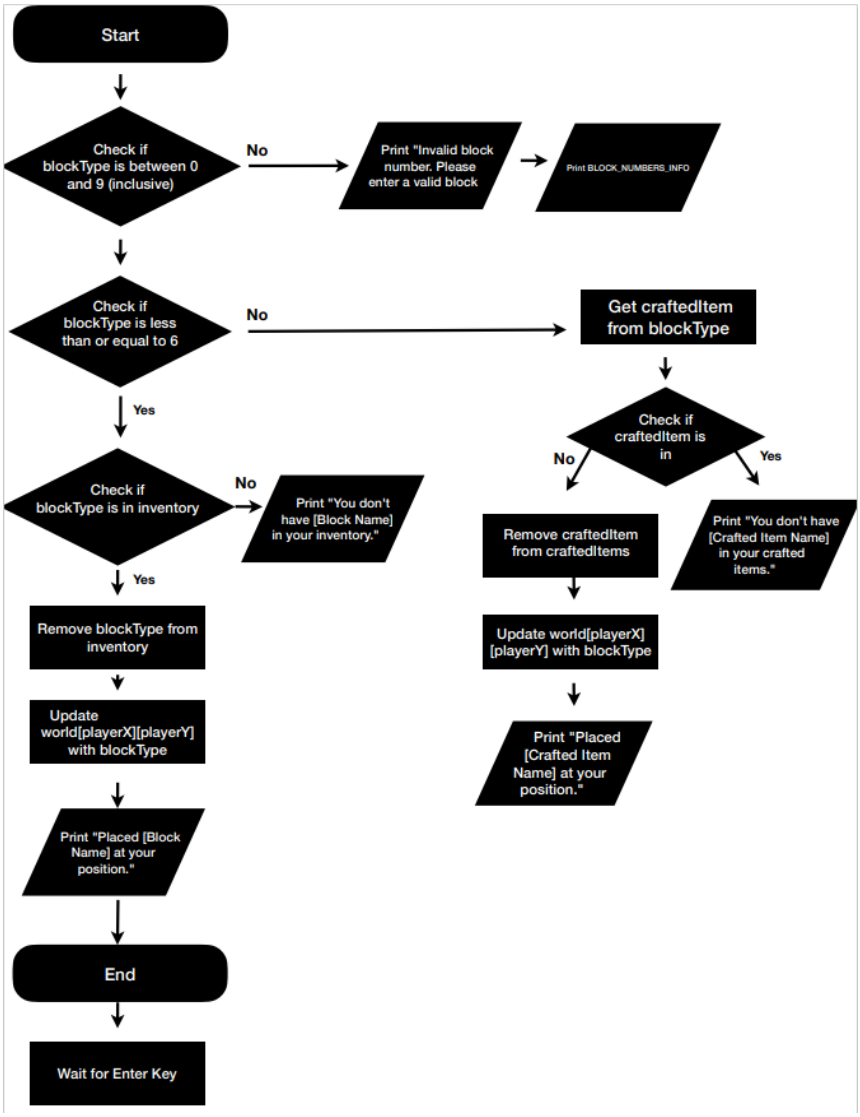
Pseudocode:

```
FUNCTION inventoryContains(item, count):
   SET itemCount TO 0
   FOR EACH itemInInventory IN inventory:
      IF itemInInventory EQUALS item THEN
         INCREMENT itemCount BY 1
         IF itemCount EQUALS count THEN
            RETURN TRUE
         END IF
      END IF
   END FOR
   RETURN FALSE
END FUNCTION
```



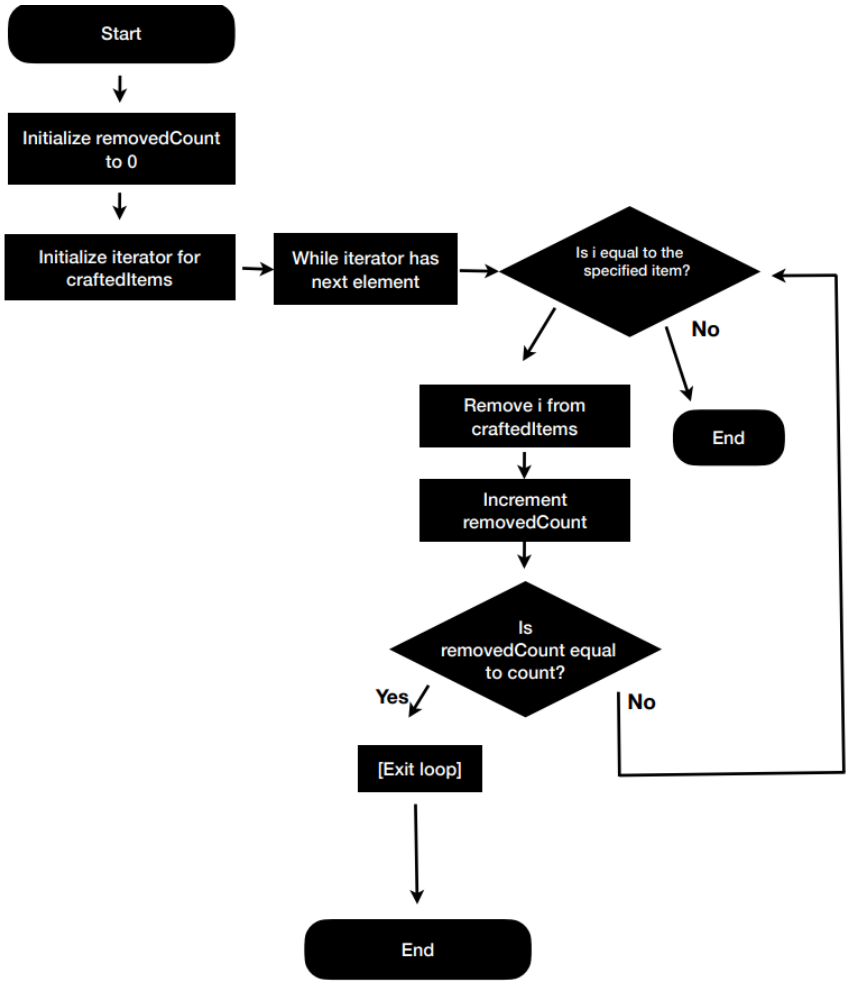
Pseudocode:

```
FUNCTION placeBlock(blockType):
    IF blockType >= 0 AND blockType <= 9 THEN
        IF blockType <= 6 THEN
            IF inventory CONTAINS blockType THEN
                CALL inventory REMOVE blockType
                SET world[playerX][playerY] TO blockType
                PRINT "Placed " + getBlockName(blockType) + " at your
position."
            ELSE
                PRINT "You don't have " + getBlockName(blockType) + " in
your inventory."
            END IF
        ELSE
            SET craftedItem TO getCraftedItemFromBlockType(blockType)
            IF craftedItems CONTAINS craftedItem THEN
                CALL craftedItems REMOVE craftedItem
                SET world[playerX][playerY] TO blockType
                PRINT "Placed " + getCraftedItemName(craftedItem) + " at
your position."
            ELSE
                PRINT "You don't have " + getCraftedItemName(craftedItem) +
" in your crafted items."
            END IF
        END IF
    ELSE
        PRINT "Invalid block number. Please enter a valid block number."
        PRINT BLOCK_NUMBERS_INFO
    END IF
    CALL waitForEnter()
END FUNCTION
```


Start
Check if blockType is between 0 and 9 (inclusive)
No
Print "Invalid block number. Please enter a valid block
Print BLOCK_NUMBERS_INFO
Check if blockType is less than or equal to 6
No
Get craftedItem from blockType
Yes
Check if craftedItem is in
No
Yes
Check if blockType is in inventory
No
Print "You don't have [Block Name] in your inventory."
Print "You don't have [Crafted Item Name] in your crafted items."
Remove craftedItem from craftedItems
Yes
Remove blockType from inventory
Update world[playerX][playerY] with blockType
Update world[playerX][playerY] with blockType
Print "Placed [Block Name] at your position."
Print "Placed [Crafted Item Name] at your position."
End
Wait for Enter Key

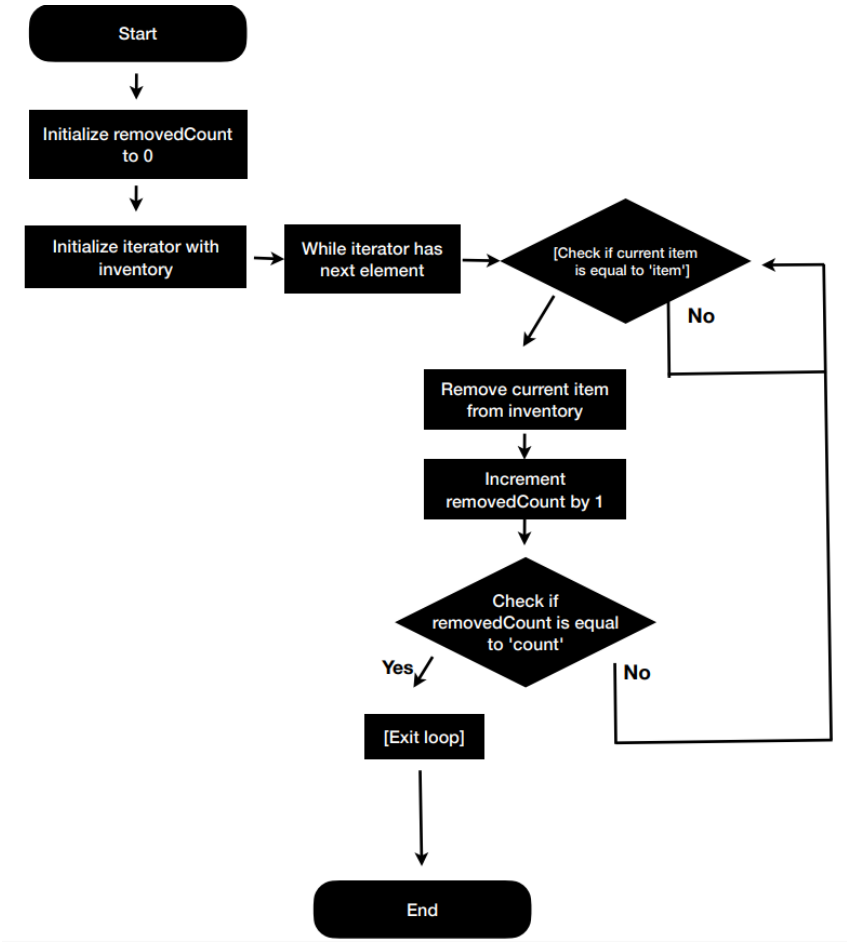
Pseudocode:

```
FUNCTION removeItemsFromCraftedItems(item, count):
    DECLARE removedCount = 0
    DECLARE iterator = craftedItems.iterator()
    WHILE iterator HAS NEXT DO
        DECLARE i = iterator.NEXT()
        IF i EQUALS item THEN
            CALL iterator.REMOVE()
            INCREMENT removedCount BY 1
            IF removedCount EQUALS count THEN
                BREAK
            END IF
        END IF
    END WHILE
END FUNCTION
```



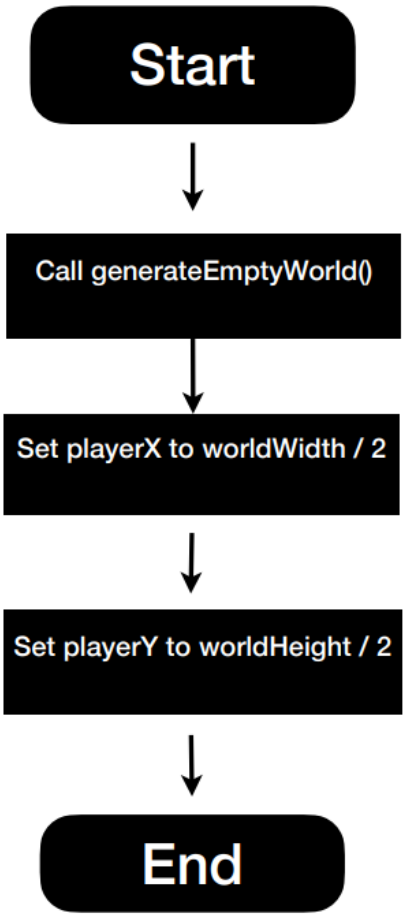
Pseudocode:

```
FUNCTION removeItemsFromInventory(item, count):
    SET removedCount TO 0
    CREATE iterator AS ITERATOR FOR inventory
    WHILE iterator HAS NEXT:
        SET i TO NEXT ITEM FROM iterator
        IF i IS EQUAL TO item THEN
            REMOVE i FROM inventory
            INCREMENT removedCount BY 1
            IF removedCount EQUALS count THEN
                EXIT LOOP
            END IF
        END IF
    END WHILE
END FUNCTION
```



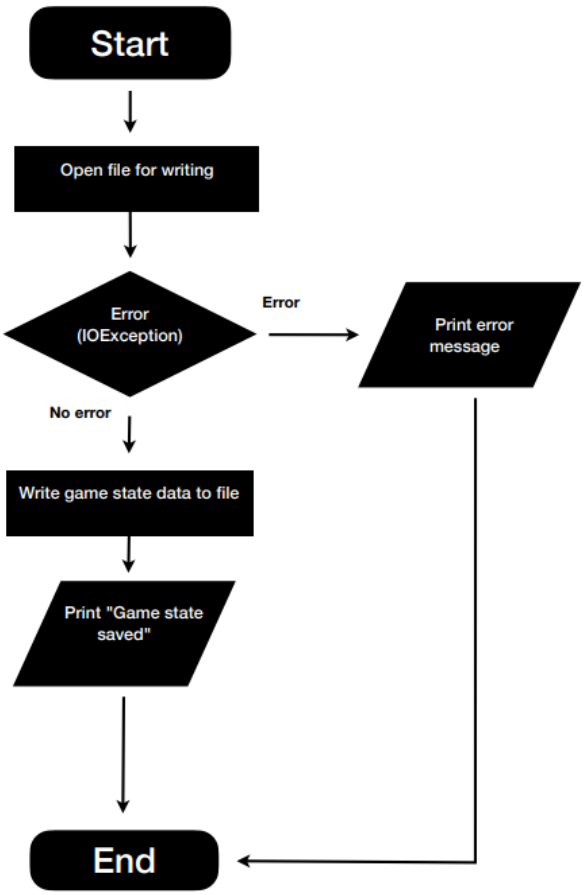
Pseudocode:

```
FUNCTION resetWorld():
    CALL generateEmptyWorld()
    SET playerX TO worldWidth / 2
    SET playerY TO worldHeight / 2
END FUNCTION
```



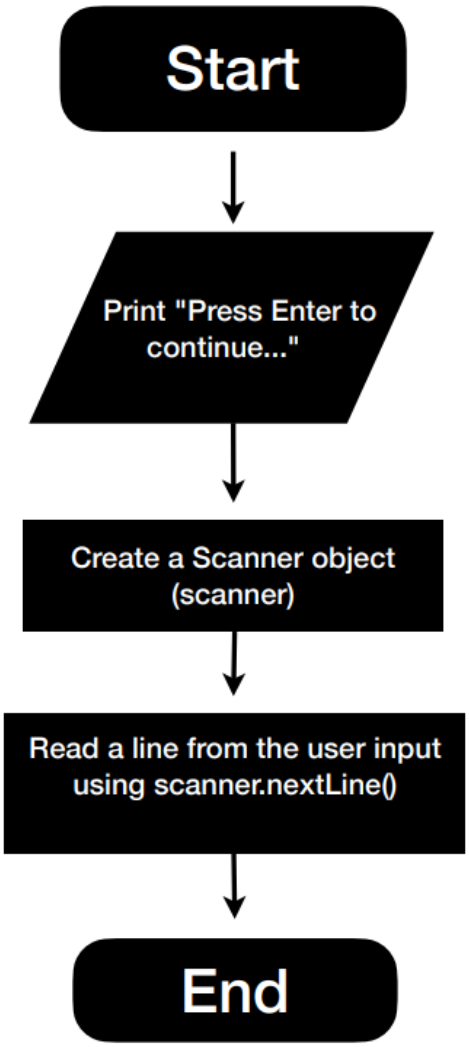
Pseudocode:

```
FUNCTION saveGame(fileName):
   TRY
      CREATE outputStream AS NEW ObjectOutputStream FOR
FileOutputStream(fileName)
      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 IOException AS e
      PRINT "Error while saving the game state: " + e.getMessage()
   END TRY
   CALL waitForEnter()
END FUNCTION
```



Pseudocode:

```
FUNCTION waitForEnter():
    PRINT "Press Enter to continue..."
    CREATE scanner AS NEW Scanner(INPUT)
    CALL scanner.nextLine()
END FUNCTION
```



10 References

1. Link - Pseudo Code Guide
2. Link - Making the flowcharts