# JavaCraft Project

BCS1110, Introduction to Computer Science


**Group 11**

| Full Name | Student ID |
|---|---|
| Long Luong | I6359380 |
| Élisa Donéa | I6356213 |
| Chris Munteanu | I6344912 |
| Alexia Raportaru | I6355814 |

Professors: Dr. Ashish Sai, Dr. Thomas Bitterman


Maastricht, October 22, 2023

# Contents

# 1 | Introduction

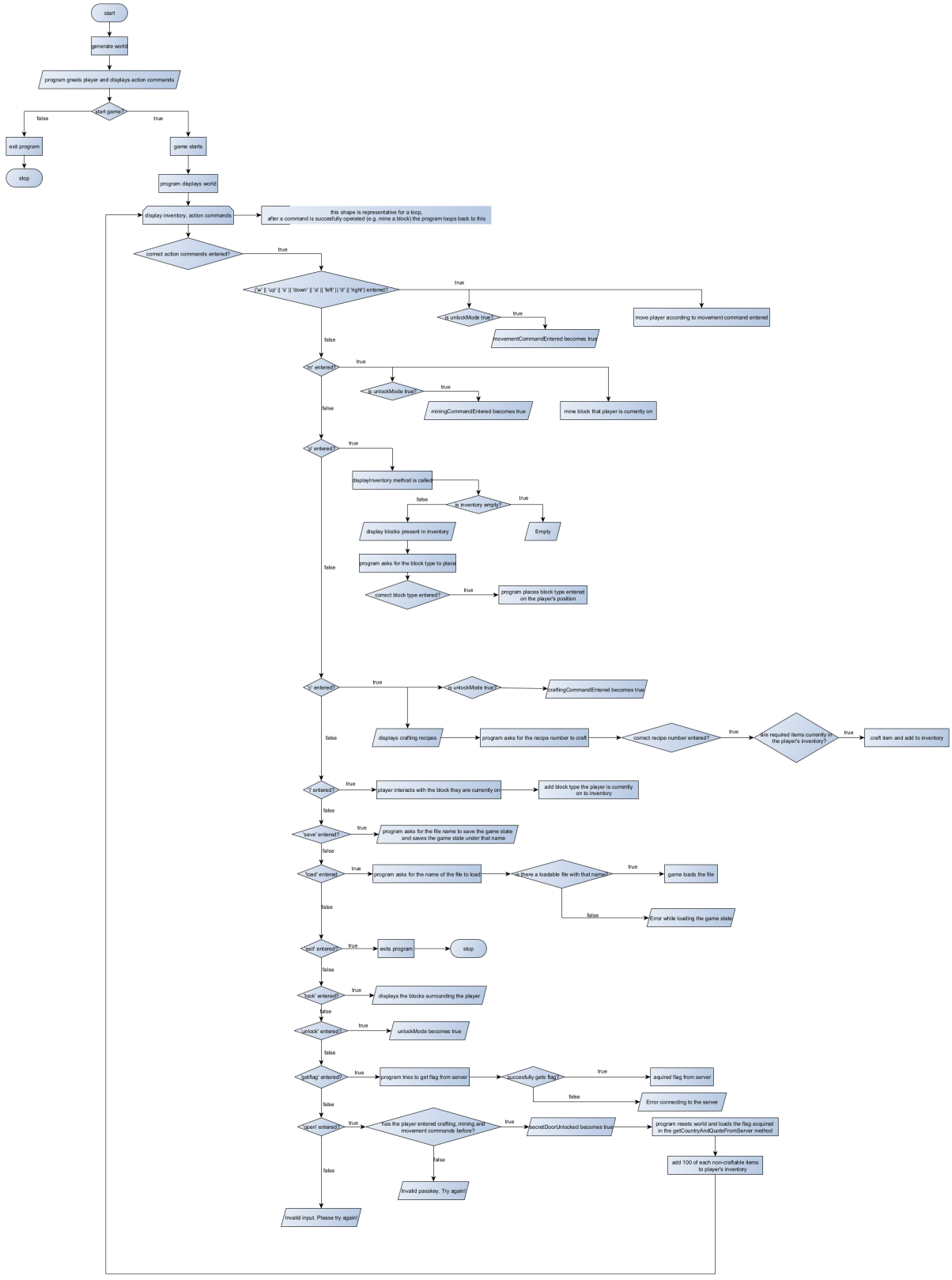
JavaCraft is a a multifaceted text-based Java game inspired by Minecraft. The game is a relatively complex Java program that brings over 35 functions to create a diverse gameplay experience. This team project is an academic exercise in computer science, logical thinking, and collaboration. Working in teams of four, we used our creativity, analytical skills, and technical skills to analyse and expand upon the existing JavaCraft game. The source code of JavaCraft can be found in the following link: https://gitlab.maastrichtuniversity.nl/bcs1110/javacraft/-/raw/main/JavaCraft.java

Below is a table that describes what each team member did on the project, divided by section:

| Section | Who did what |
|---|---|
| Introduction | Long (100%) |
| JavaCraft's Workflow | Alexia (100%) |
| Functionality Exploration | Long (60%), Élisa (40%) |
| Secret Door FSA Design | Élisa (50%), Chris (50%) |
| Git Collaboration & Version Control | Alexia (25%), Élisa (25%), Chris (25%), Long (25%) |
| Extending the Game Code | Alexia (35%), Long (65%) |
| Interacting with Flags API | Élisa (50%), Chris (15%), Long (35%) |
| Report | Long (100%) |

# 2 | JavaCraft's Workflow

In order to fully understand the mechanism of the game, a flowchart of the entire game is provided below. The flowchart is accompanied by pseudocode.




```
generate world
print welcome message and instructions
```

```
ask player if they want to start the game
if choice is equal to 'y' then start game
   clear screen
   display world

   loop
   display inventory, display legend

   if input is 'wasd' then move player accordingly
      if unlockMode is true then movementCommandEntered becomes true

   else if input is 'm' then mine the block the player is currently standing on
      if unlockMode is true then miningCommandEntered becomes true

   else if input is 'p' then function displayInventory and ask player which block they want
       to place                                    if inventory is empty display "Empty"
    else display inventory
      if input is correct then place block on the player's current position
      else inform player they do not have that specific block in their inventory

   else if input is 'c' then display crafting recipes
      ask for recipe number in order to craft
          if input is correct then check whether the player has the items necessary for the
              crafting recipe
             if the player has the necessary items then proceed crafting and add the item to
                 the player's inventory
             else inform the player they have insufficient resources to craft
      if unlockMode is true then craftinCommandEntered becomes true

   else if input is 'i' then player interacts with the block they are currently standing on
      if the block they are currently on is not air then add the item to their inventory

   else if input is 'save' then ask player for a name for the current state file they want
       to save
      save file

   else if input is 'load' then ask player for the name of the file they want to load
      if there is a file with that name then load file
      else inform player there has been an error loading the game state

   else if input is 'exit' then exit program

   else if input is 'look' then inform the player of the block types surrounding them

   else if input is 'unlock' then unlockMode becomes true

   else if input is 'getflag' then try to get flag from the server
      if unsuccesful then inform the player that there has been an error connecting to the
          server

   else if input is 'open' then check if the booleans craftingCommandEntered,
       movementCommandEntered, miningCommandEntered true
      if they are true then secretDoorUnlocked becomes true
         reset world
         load flag world
         add 100 of leaves, stone, iron ore and wood to player's inventory

   else inform player that their input is invalid
end if
else print "Game not started. Goodbye!" and exit program
end if
```

# 3 | Functionality Exploration

This section describes several functions found in the JavaCraft.java file. Out of the functions, there are fifteen who have a flowchart and pseudocode.

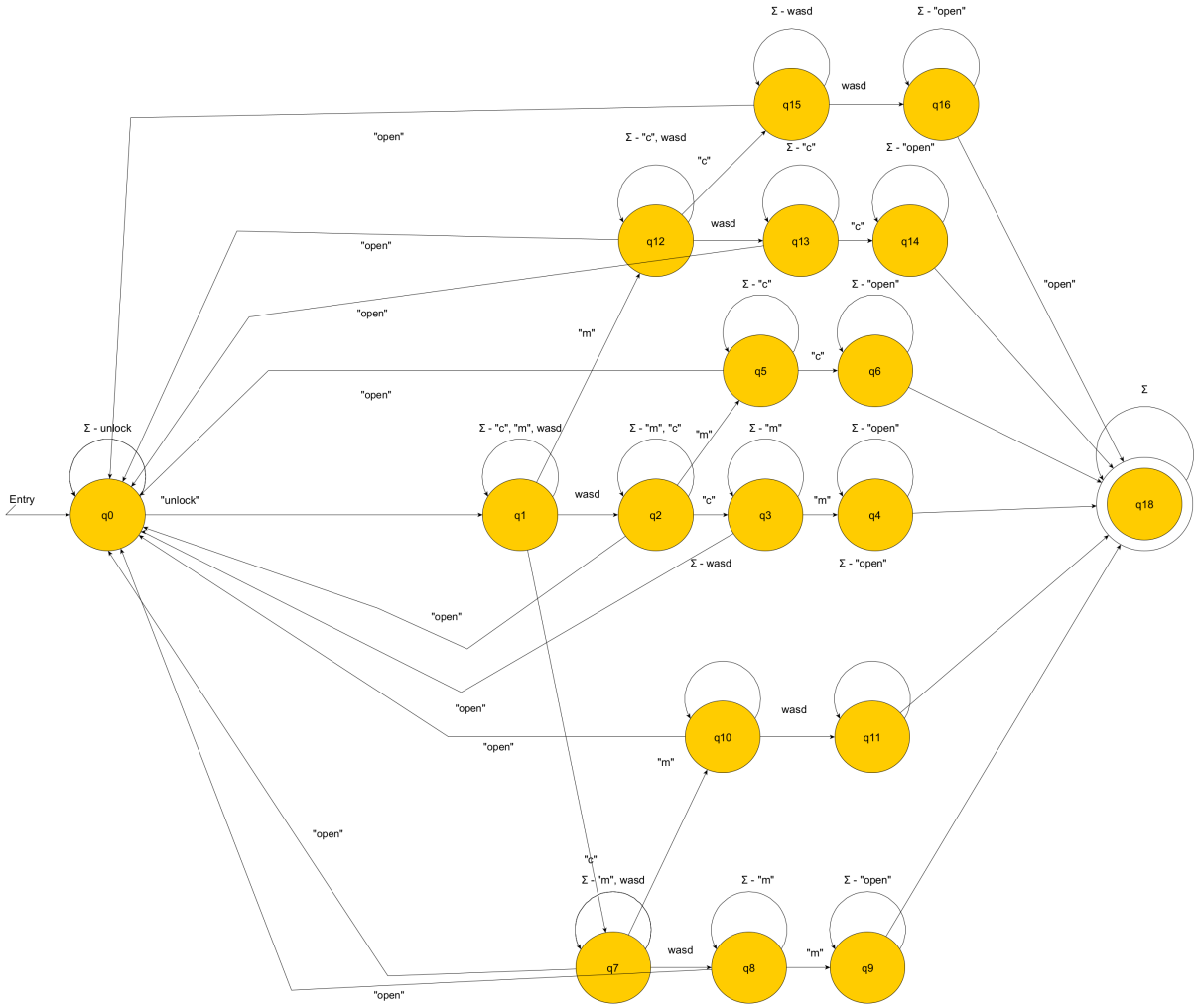
**Table 3.1:** A table that describes functions used in javacraft

| No. | Function Name | Description |
|---|---|---|
| 1 | `void generateWorld` | assigns integer to every tile of the world |
| 2 | `void initGame` | creates world with width `worldWidth` and height `worldHeight` |
| 3 | `void main` | main function |
| 4 | `void startGame` | starts the game |
| 5 | `void movePlayer` | moves player horizontally or vertically |
| 6 | `void mineBlock` | mines block player is on if block is not air |
| 7 | `String getBlockSymbol` | returns symbol of `blockType` |
| 8 | `void resetWorld` | clears the world and sets player position in middle |
| 9 | `void generateEmptyWorld` | generates an empty world |
| 10 | `void clearScreen` | clears terminal |
| 11 | `void lookAround` | prints out adjacent squares to player |
| 12 | `void fillInventory` | completely fills up inventory of player |
| 13 | `void displayLegend` | displays a legend of what each tile represents |
| 14 | `void displayWorld` | prints out all tiles of the world |
| 15 | `void displayInventory` | prints out obtained items & crafted items |
| 16 | `void loadGame` | loads the game from file `fileName` |
| 17 | `void saveGame` | saves the game in file `fileName` |
| 18 | `void interactWithWorld` | interacts with item player is standing on |
| 19 | `void addCraftedItem` | adds item `craftedItem` to array `craftedItems` |
| 20 | `void removeItemsFromInventory` | removes item `item` `count` times from inventory |
| 21 | `boolean inventoryContains` | returns boolean of whether `item` is in inventory |
| 22 | `void placeBlock` | places block `blockType` at player position |
| 23 | `void displayCraftingRecipes` | prints out available crafting recipes |
| 24 | `void craftItem` | crafts an item based on argument `recipe` |
| 25 | `void craftIronIngot` | crafts an iron ingot |
| 26 | `void craftStick` | crafts a stick |
| 27 | `void waitForEnter` | waits for operator to press Enter |
| 28 | `String getBlockTypeFromCraftedItem` | returns integer of `craftedItem` |
| 29 | `String getCraftedItemFromBlockType` | returns integer of `blockType` |
| 30 | `String getBlockName` | returns name of `blockType` |
| 31 | `String getBlockColor` | returns color of `blockType` |
| 32 | `String getCraftedItemName` | returns name of `craftedItem` |
| 33 | `String getCraftedItemColor` | returns color of `craftedItem` |
| 34 | `char getBlockChar` | returns char of `blockType` |
| 35 | `void craftWoodenPlanks` | crafts wooden planks |
| 36 | `void getCountryAndQuoteFromServer` | makes HTTP request and writes data to server and prints country and quote |

# 4 | Finite State Automata (FSA) Design

The following describes the Secret Door Logic through an FSA illustration and a description:

We have 18 finite set of states and in our alphabet we have ”open” , ”wasd”, ”c”, ”m” and ”unlock” On step one Q0 you have to write unlock if not you will remain in Q0 After you have arrived in Q1 you have the choice in between wasd, c, m because in able to unlock the secret door you must c, craft then m, mine or wasd to move in any order that’s why after Q1 it divides in 3 because the order is not important. If you type a move command (w,a,s or d) you enter Q2, the only options you have from now are mine or craft if not you stay in Q2 unless you type open which sends you back to Q0 and have to start again. If you type c you go into Q3 so the last thing you have to write is mine if you write anything else you stay in Q3 expect ”open” which sends you in Q0 after mining you enter Q4. From there your only option is to ”open” which unlocks the secret door. Now because the order doesn’t matter after Q2 you can also type m for mine which sends you in Q5 where the only option is c, ”craft” if you type anything else you go back to Q0 after you type C you entre Q6 where you can only type ”open” which will unlock the secret door. We have created 2 other branches based on the same principal so we can cover all the possible ways you can access the secret door
Note: in our alphabet ”wasd” stands for ”w”, ”a”, ”s” or ”d”



# 5 | Git Collaboration & Version Control

The link to the JavaCraft branch can be found here: https://gitlab.maastrichtuniversity.nl/bcs1110/javacraft/-/tree/group11

All the files of the project can be found in the repository.
Everything was done on one branch called group11. There were no conflicts during the process due to the way we divided the tasks. Each section of the report is done at most by two different people, with the exception of the 15 flowcharts with pseudocode.
For the 15 flowcharts with pseudocode, we created two folders in the git branch: one for flowcharts and one for pseudocode. The folder contains one file for each function, meaning that in total we have 15 (or more) files per folder. Having everything separated in folders makes it easy to add the files to this document and to also keep track of each other's progress.

# 6 | Extending the Game Code

We extended the code by creating a crafting recipe for a cake. The recipe involves 3 milk buckets, 3 wheat, 2 sugar and 1 egg. Upon crafting the cake the player gets a cake and 3 empty buckets. The cake recipe in JavaCraft is the same cake recipe in Minecraft. In order to be able to craft a cake, we have added the following blocks:

1. Cow
2. Chicken
3. Wheat
4. Sugar Cane

The world generation has been modified to include the four aforementioned blocks. In particular, the method `generateWorld` has been modified.

```
public static void generateWorld() {
    Random rand = new Random();
    for (int y = 0; y < worldHeight; y++) {
      for (int x = 0; x < worldWidth; x++) {
        int randValue = rand.nextInt(100);
        if (randValue < 10) {
          world[x][y] = WOOD;
        } else if (randValue < 20) {
          world[x][y] = LEAVES;
        } else if (randValue < 28) {
          world[x][y] = WHEAT;
        } else if (randValue < 37) {
          world[x][y] = CHICKEN;
        } else if (randValue < 45) {
          world[x][y] = STONE;
        } else if (randValue < 60) {
          world[x][y] = IRON_ORE;
        } else if (randValue < 65) {
          world[x][y] = COW;
        } else if (randValue < 70) {
          world[x][y] = SUGARCANE;
        } else {
          world[x][y] = AIR;
        }
      }
    }
}
```

In addition, we have implemented new items:

1. Milk Bucket
2. Egg
3. Empty Bucket
4. Sugar
5. Cake

The recipes for the aforementioned items (except for the milk bucket and egg) each have their own methods: `craftBucket`, `craftCake`, and `craftSugar`. Their methods are as follows:

```
public static void craftBucket() {
    if (craftedItemsContains(CRAFTED_IRON_INGOT, 3)) {
      removeItemsFromCraftedItems(CRAFTED_IRON_INGOT, 3);
      addCraftedItem(CRAFTED_BUCKET);
```

```
      System.out.println("Crafted Bucket.");
    } else {
      System.out.println("Insufficient resources to craft Bucket.");
    }
}
public static void craftCake() {
    boolean cond = craftedItemsContains(CRAFTED_SUGAR, 2) && inventoryContains(EGG, 1) &&
        inventoryContains(WHEAT, 3) && inventoryContains(MILK_BUCKET, 3);
    if (cond) {
      removeItemsFromCraftedItems(CRAFTED_SUGAR, 2);
      removeItemsFromInventory(EGG, 1);
      removeItemsFromInventory(WHEAT, 3);
      removeItemsFromInventory(MILK_BUCKET, 3);
      addCraftedItem(CRAFTED_BUCKET);
      addCraftedItem(CRAFTED_BUCKET);
      addCraftedItem(CRAFTED_BUCKET);
      addCraftedItem(CRAFTED_CAKE);
      System.out.println("Crafted Cake.");
    } else {
      System.out.println("Insufficient resources to craft Cake.");
    }
}
public static void craftSugar() {
    if (inventoryContains(SUGARCANE, 1)) {
      removeItemsFromInventory(SUGARCANE, 1);
      addCraftedItem(CRAFTED_SUGAR);
      System.out.println("Crafted Sugar.");
    } else {
      System.out.println("Insufficient resources to craft Sugar.");
    }
}
```

Note that we call for the methods `craftedItemsContains` and `removeItemsFromCraftedItems`, which were not in the original code. This is done because items such as the iron bucket and sugar are stored in `craftedItems`, which is separate from the inventory `inventory`. The method is similar to the existing methods `inventoryContains` and `removeItemsFromInventory` with the only difference being which inventory the method accesses. The methods `craftedItemsContains` and `removeItemsFromCraftedItems` are as follows:

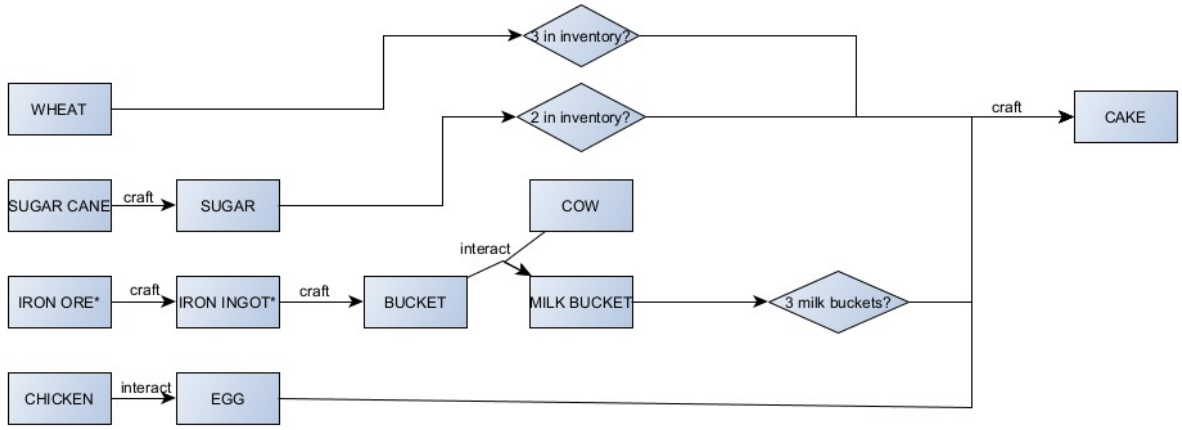
```
    public static boolean craftedItemsContains(int item) {
    return craftedItems.contains(item);
  }
    public static boolean craftedItemsContains(int item, int count) {
    int itemCount = 0;
    for (int i : craftedItems) {
      if (i == item) {
        itemCount++;
        if (itemCount == count) {
          return true;
        }
      }
    }
    return false;
  }
  public static void removeItemsFromCraftedItems(int item, int count) {
    int removedCount = 0;
    Iterator<Integer> iterator = craftedItems.iterator();
    while (iterator.hasNext()) {
      int i = iterator.next();
      if (i == item) {
        iterator.remove();
        removedCount++;
        if (removedCount == count) {
```

```
        break;
      }
    }
  }
}
```

The Milk Bucket can be obtained if the player interacts with a cow while the player has at least one bucket in their inventory. The Egg can be obtained when interacting with the chicken. Both the cow and the chicken cannot be mined as it does not make much sense to mine an animal. Upon trying to mine either animal, the message `Why would you want to mine an animal? D:` appears. For this to work the method `mineBlock` has been modified.

```
  public static void mineBlock() {
  int blockType = world[playerX][playerY];
  if (blockType != AIR && blockType != COW && blockType != CHICKEN) {
    inventory.add(blockType);
    world[playerX][playerY] = AIR;
    System.out.println("Mined " + getBlockName(blockType) + ".");
  } else if (blockType == AIR) {
    System.out.println("Cannot mine air.");
  } else {
    System.out.println("Why would you want to mine an animal? D:");
  }
  waitForEnter();
}
```

The flowchart on how to craft a cake can be found below:



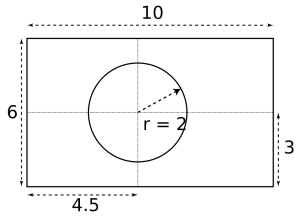
# 7 | Interacting with Flags API

The method `getCountryAndQuoteFromServer` contains everything related to the flags API. This method is run when you are playing JavaCraft and the command `getflag` is typed. The method initially did not contain and json or a valid url, which has been changed. The updated part of the code can be found below. Note that only the modified parts of the code are included in the method: the unmodified parts are replaced with ellipses surrounded by brackets:

```
  public static void getCountryAndQuoteFromServer() {
  try {
    String link = "https://flag.ashish.nl/get_flag";
    URL url = new URL(link);
    [...]
    String payload = "{\n" +
       "    \"group_number\": \"11\",\n" +
       "    \"group_name\": \"group11\",\n" +
       "    \"difficulty_level\": \"hard\"\n" +
       "}";
    [...]
  }
}
```

The flags API and documentation can be found at https://flag.ashish.nl/. The POST get_flag method requires json that has 3 parameters: `group_number`, `group_name` and `difficulty_level`. When a successful call has been made, the method `getCountryAndQuoteFromServer` will print a country's name and a quote. In the case of our group `group11`, the chosen difficulty level was `hard` and the flag we got is the flag of Bangladesh.

The flag of Bangladesh is a dark green flag with a red circle. The flag is constructed as follows:




Based on a given width and height, we can determine the center of the circle as well as the radius. Let $x$ and $y$ be the width and height of the world respectively. Let $x_r$ and $y_r$ be the x and y coordinate of the circle. Let $r$ be the radius of the circle. Using the ratios of the flag we can conclude that $x_r = x \cdot \frac{4.5}{10}$ and $y_r = y \cdot \frac{1}{2}$ and $r = \frac{2}{10}x = \frac{2}{6}y$. The formula of the circle goes as follows:

$$(x - x_r)^2 + (y - y_r)^2 = r^2$$

We now loop through every coordinate of the flag. If the condition $(x - x_r)^2 + (y - y_r)^2 \leq r^2$ is true it means that at that given x and y coordinate the coordinates are inside of the circle. We then render the square to be red. If the condition is not met then the square is green.

```
private static void generateEmptyWorld() {
  world = new int[NEW_WORLD_WIDTH][NEW_WORLD_HEIGHT];
  int xCircleCenter = (NEW_WORLD_WIDTH * 9/20);
  int yCircleCenter = NEW_WORLD_HEIGHT / 2;
  int r = (NEW_WORLD_WIDTH / 10 * 2);
   for (int y = 0; y < NEW_WORLD_HEIGHT; y++) {
    for (int x = 0; x < NEW_WORLD_WIDTH; x++) {
      if ((x-xCircleCenter)*(x-xCircleCenter)+(y-yCircleCenter)*(y-yCircleCenter) <= r*r) {
        world[x][y] = REDBLOCK;
      } else { world[x][y] = GREENBLOCK; }
    }
  }
}
```

# 8 | References

[1] mathisfun. Circle equations. https://www.mathsisfun.com/algebra/circle-equations.html.

[2] Programiz. Flowchart in programming. https://www.programiz.com/article/flowchart-programming.

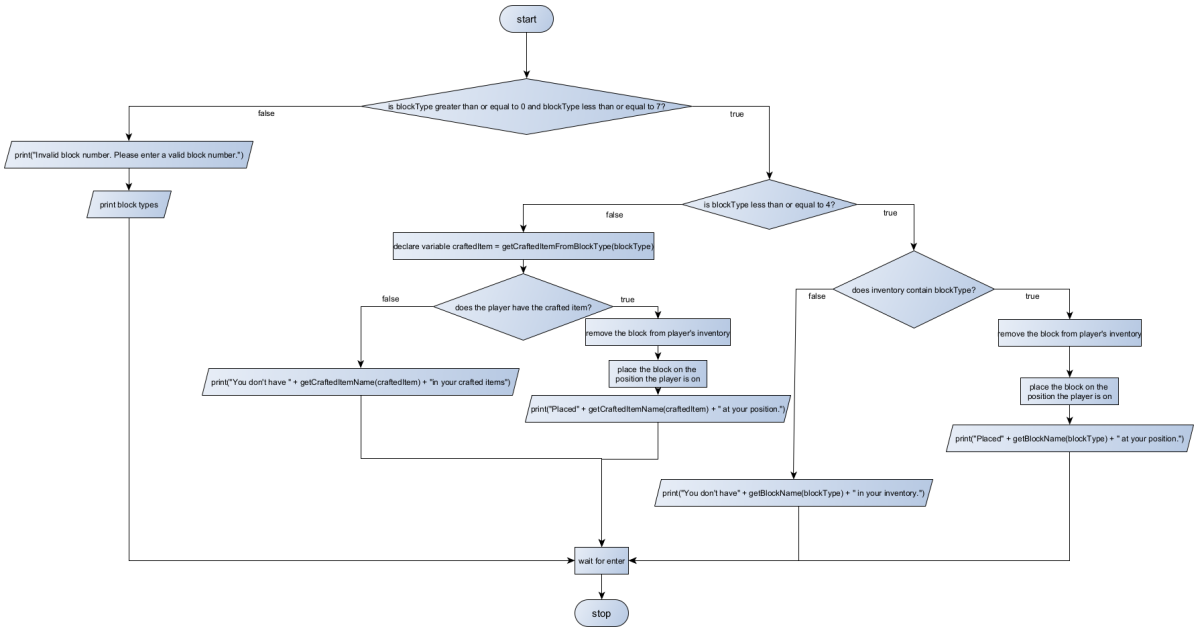
[3] Ashish Sai. Introduction to computer science. https://bcs1110.ashish.nl/.

[4] Ashish Sai. Javacraft. https://gitlab.maastrichtuniversity.nl/bcs1110/javacraft/.

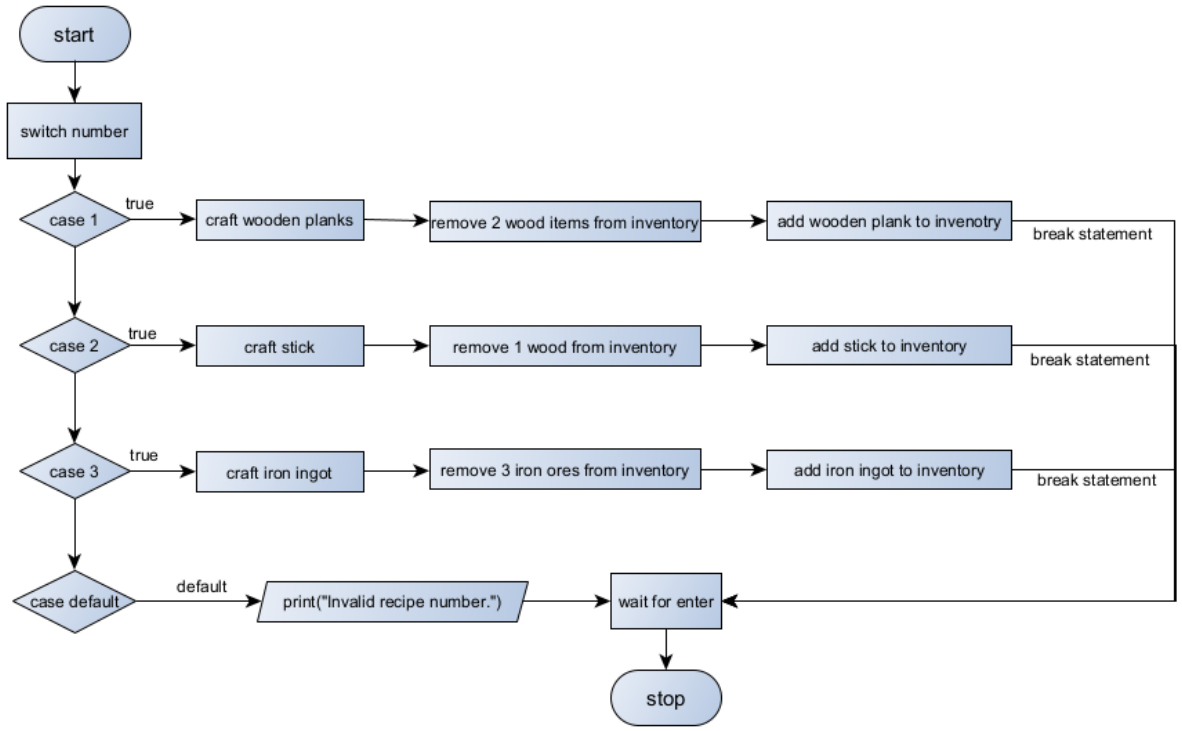
[5] Wikipedia. Flag of bangladesh. https://en.wikipedia.org/wiki/Flag_of_Bangladesh.

# A | Appendix: pseudocode and flowcharts

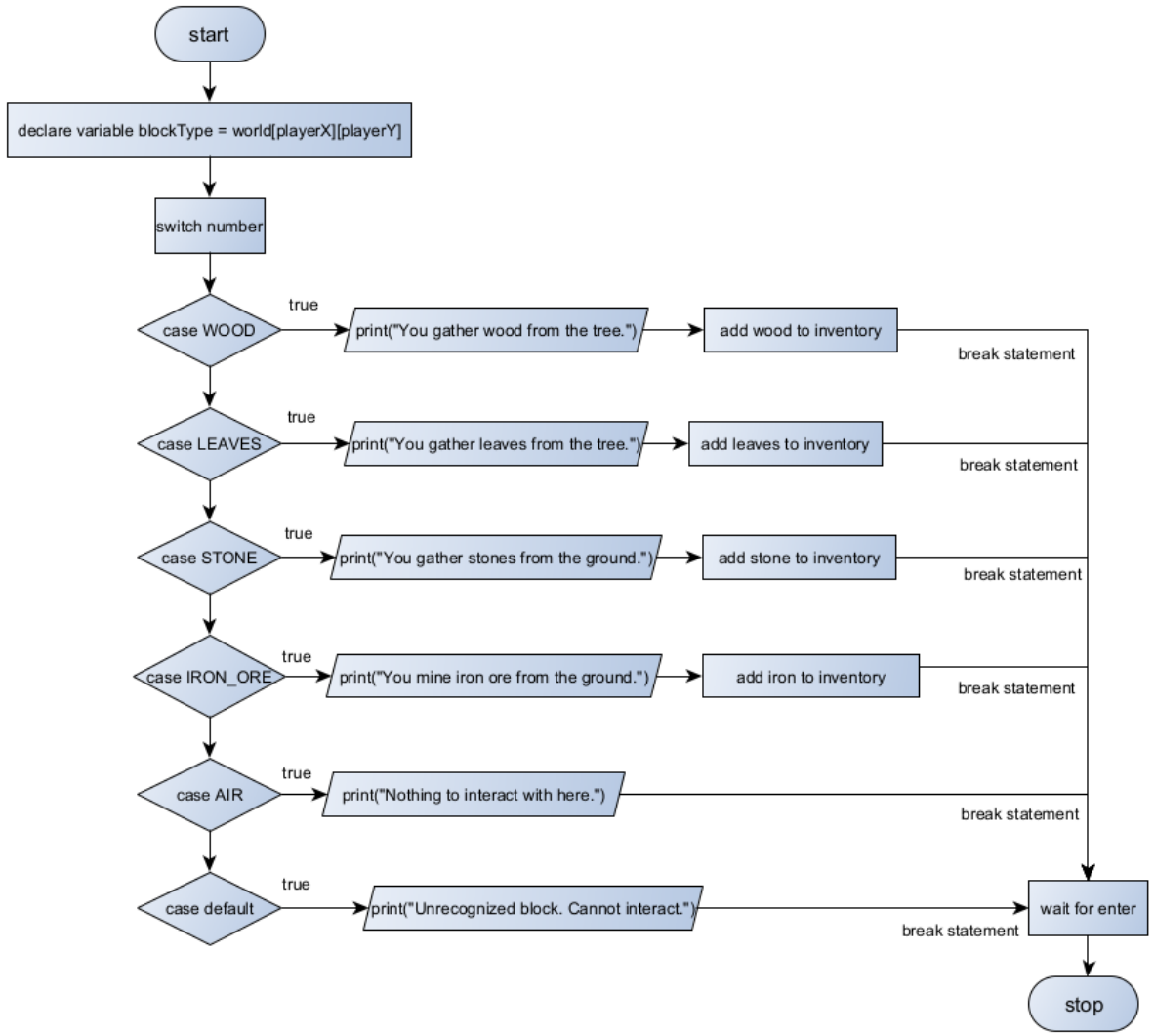
This appendix provides the full blocks of pseudocode and its flowcharts.




```
function placeBlock(blockType)

if blockType is greater than or equal to 0 and blockType is less than or equal to 7 then
   if blockType less than or equal to 4 then
      if inventory contains blockType then
         inventory.remove(blockType)
         world[playerX][playerY] = blockType
         print("Placed" + getBlockName(blockType) + " at your position.")

      else print("You don't have " + getBlockName(blockType) + " in your inventory")
      end if
   else
      craftedItem = getCraftedItemFromBlockType(blockType)
      if craftedItems contains craftedItem then
         craftedItems.remove(craftedItem)
         world[playerX][playerY] = blockType
         print("Placed" + getCraftedItemName(craftedItem)+ " at your position.")
      else print("You don't have " + getCraftedItemName() + " in your inventory.")
      end if
   end if
else print("Invalid block number. Please enter a valid block number.")
print(BLOCK_NUMBERS_INFO)
end if
waitForEnter()
end function
```

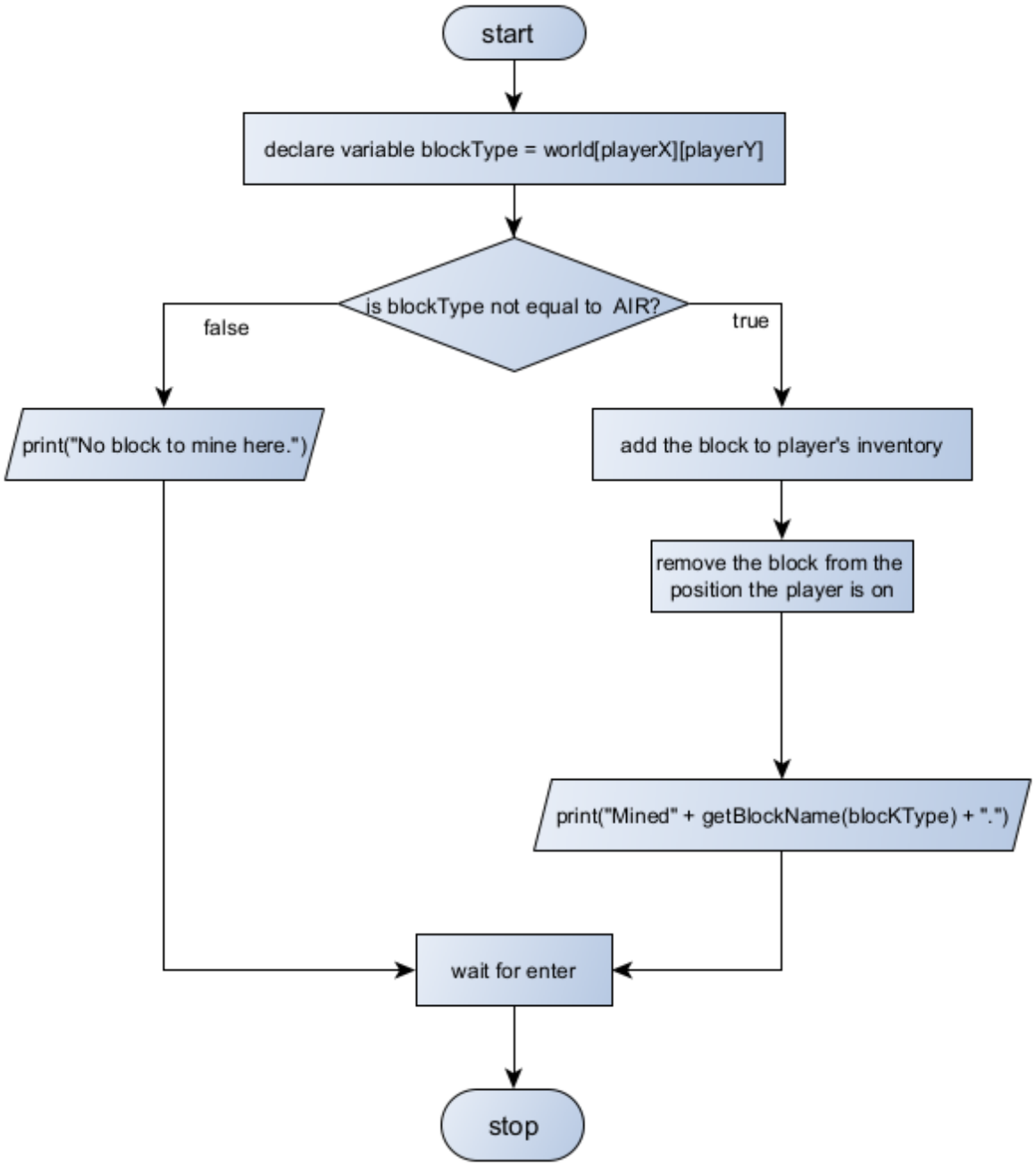



```
function craftItem(recipe)

switch(recipe)
case 1:
   craftWoodenPlanks()
end if
case2:
   craftStick()
end if
case3:
   craftIronIngot()
end if
default:
   print("Invalid recipe number.")
end switch
waitForEnter()
end function
```

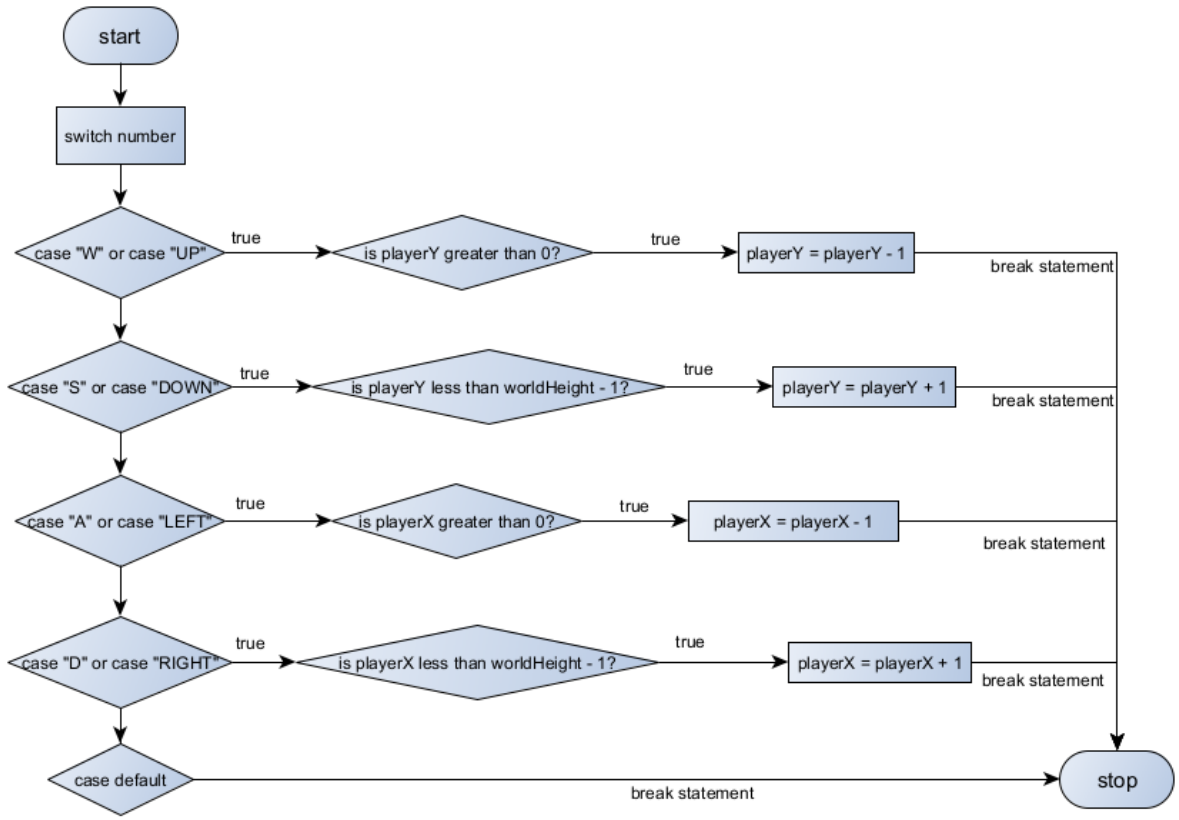



```
function interactWithWorld()

blockType = world[playerX][playerY]
switch(blockType):
case WOOD:
   print("You gather wood from the tree.")
   inventory.add(WOOD)
end if
case LEAVES:
   print("You gather leaves from the tree.")
   inventory.add(LEAVES)
end if
case STONE:
   print("You gather stones from the ground.")
   inventory.add(STONE)
end if
case IRON_ORE:
   print("You mine iron ore from the ground.")
   inventory.add(IRON_ORE)
end if
case AIR:
   print("Nothing to interact with here.")
end if
default: print("Unrecognized block. Cannot interact.")
end switch
```

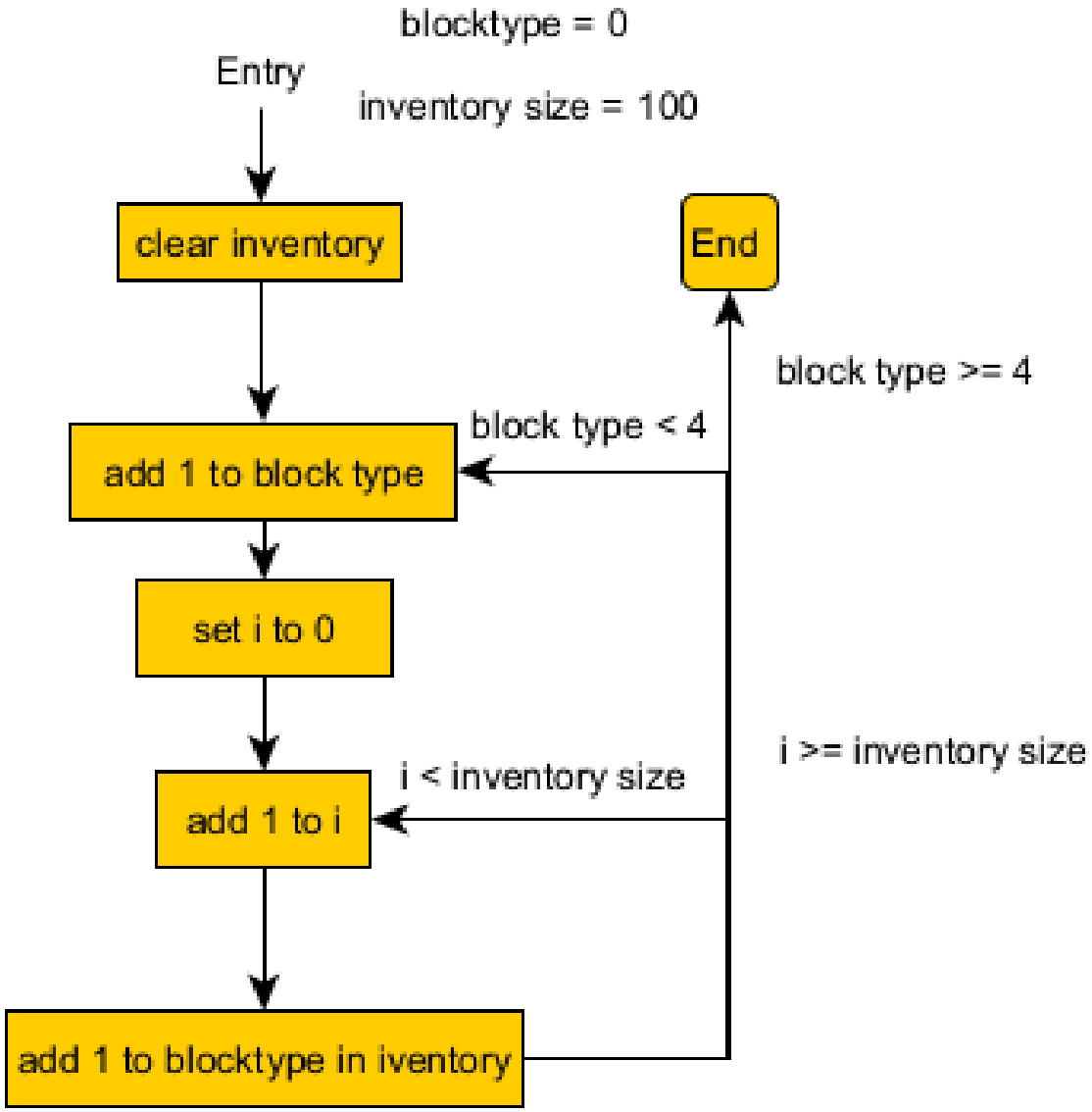
```
waitForEnter()
end function
```



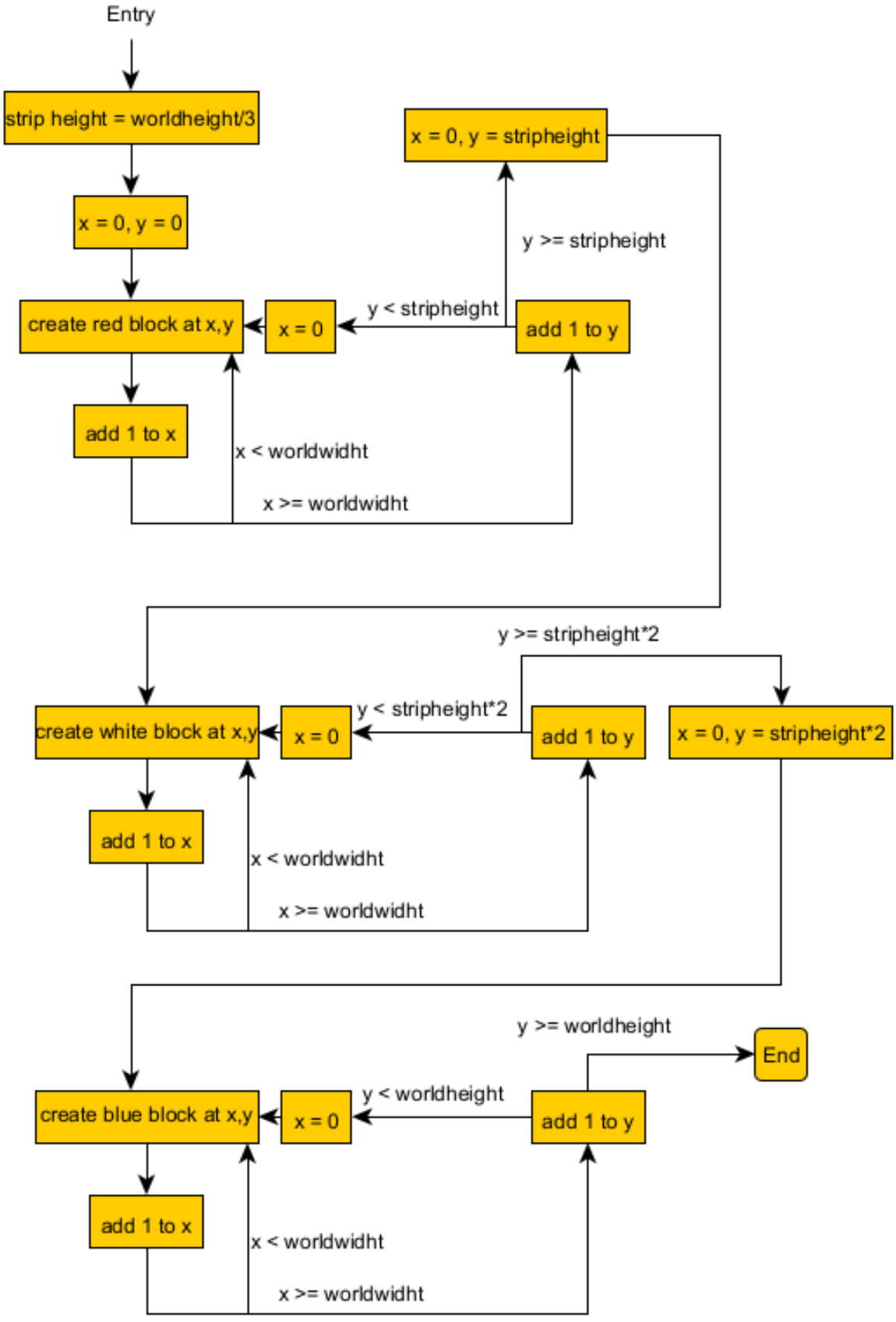

```
function mineBlock()

blockType = world[playerX][playerY]
if blockType is not equal to AIR then
   inventory.add(blockType)
   world[playerX][playerY] = AIR
   print("Mined " + getBlockName(blockType) + ".")
else print("No block to mine here.")
end if
waitForEnter()
end function
```




```
function movePlayer(direction):

direction = uppercase(direction) //converts direction to uppercase for consistency
switch(direction):
case "W" or "UP":
   if playerY > 0 then playerY = playerY - 1
   end if
case "S" or "DOWN":
   if playerY < worldHeight - 1 then playerY = playerY + 1
   end if
case "A" or "LEFT":
   if playerX > 0 then playerX = playerX - 1
   end if
case "D" or "RIGHT":
   if playerX < worldWidth - 1 then playerX = playerX + 1
   end if
default: //do nothing
end switch
end function
```

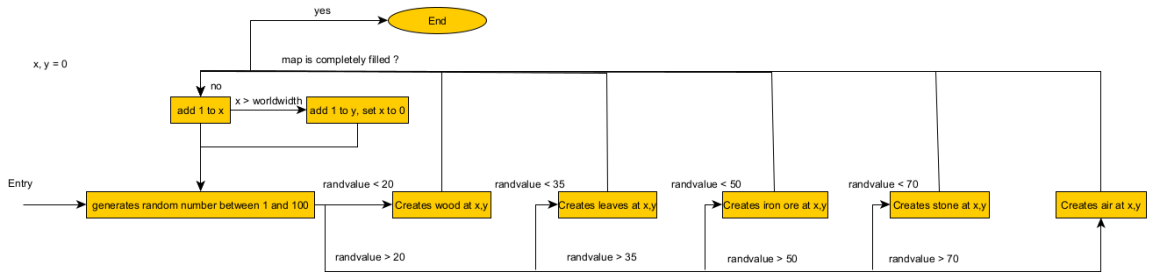



```
fillInventory()
Set INVENTORY_SIZE = 100

Clear inventory
FOR blockType = 0; blockType <= 4; blockType ++:
   FOR i = 0; i < INVENTORY_SIZE; i++:
      add block blockType to inventory

end function
```



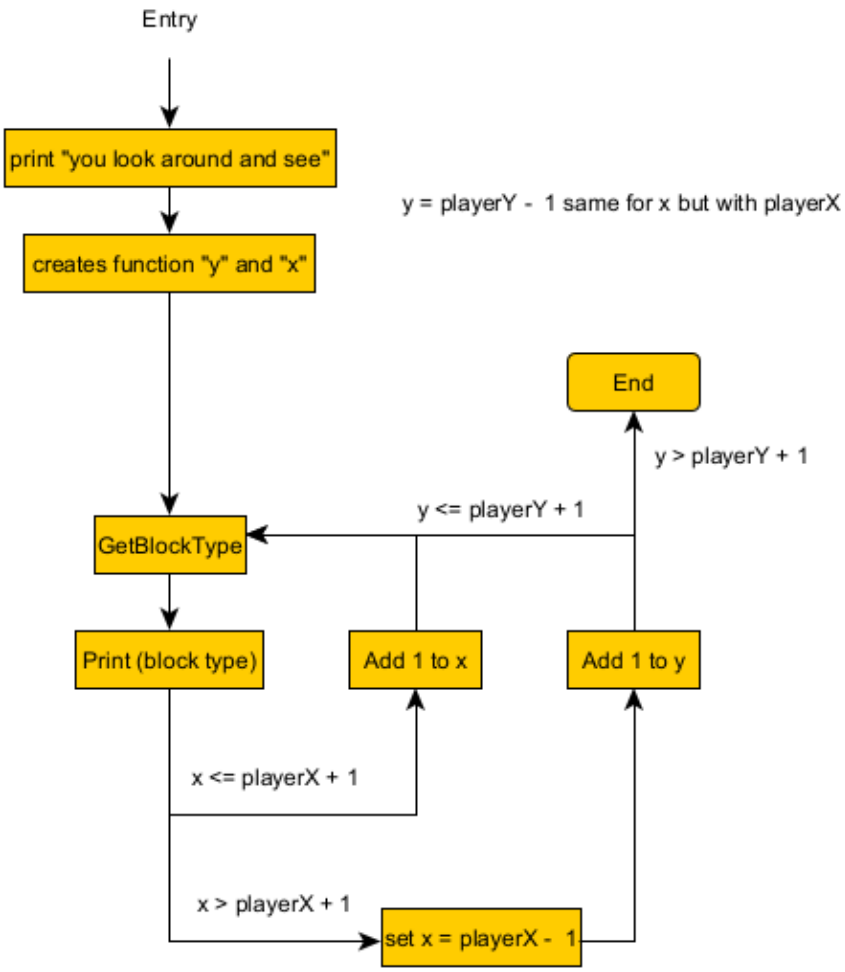

```
function generateEmptyWorld()
```

```
Set 2d array world = new int[NEW_WORLD_WIDTH][NEW_WORLD_HEIGHT]
Set redBlock = 1
Set whiteBlock = 4
Set blueBlock = 3
Set stripHeight = NEW_WORLD_HEIGHT / 3

FOR y = 0; y < stripHeight; y++:
    FOR x = 0; x < NEW_WORLD_WIDTH; x++:
        world[x][y] = redBlock

FOR y = stripHeight; y < stripHeight * 2; y++:
    FOR x = 0; x < NEW_WORLD_WIDTH; x++:
        world[x][y] = whiteBlock

FOR y = stripHeight * 2; y < NEW_WORLD_HEIGHT; y++:
    FOR x = 0; x < NEW_WORLD_WIDTH; x++:
        world[x][y] = blueBlock
end function
```

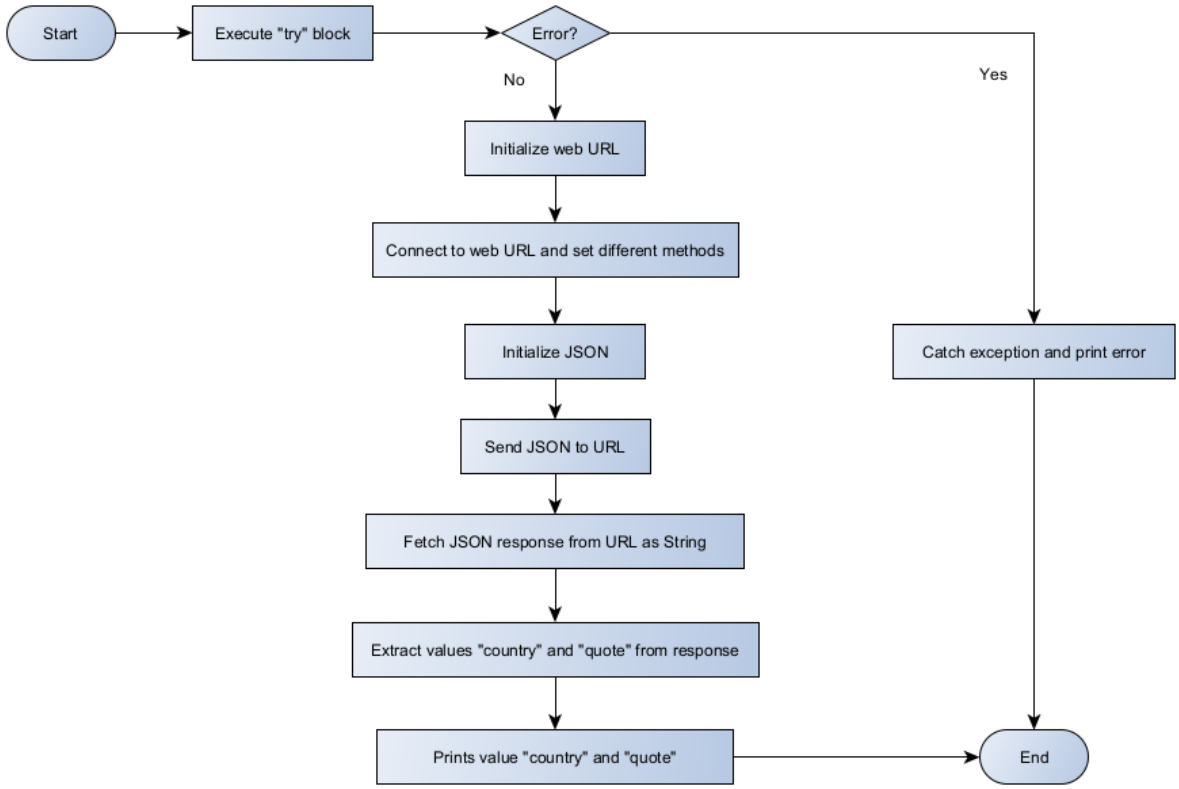



```
function generateWorld()

FOR y = 0; y < WORLD_HEIGHT; y++:
  FOR x = 0; x < WORLD_WIDTH; x++:
    creates random number between 1 and 100
    if random number < 20
      creates wood at x, y
    else if random number < 35
      creates leaves at x, y
    else if random number < 50
      creates stone at x, y
    else if random number < 20
      creates iron ore at x, y
   else create air at x, y
end function
```


```
function initGame()

Set worldwidth
Set worldheight
Set world = [worldwidht][worldheight]
Set playerx = worldwidght / 2
Set playery = worldheight / 2
Creates array list inventory
```

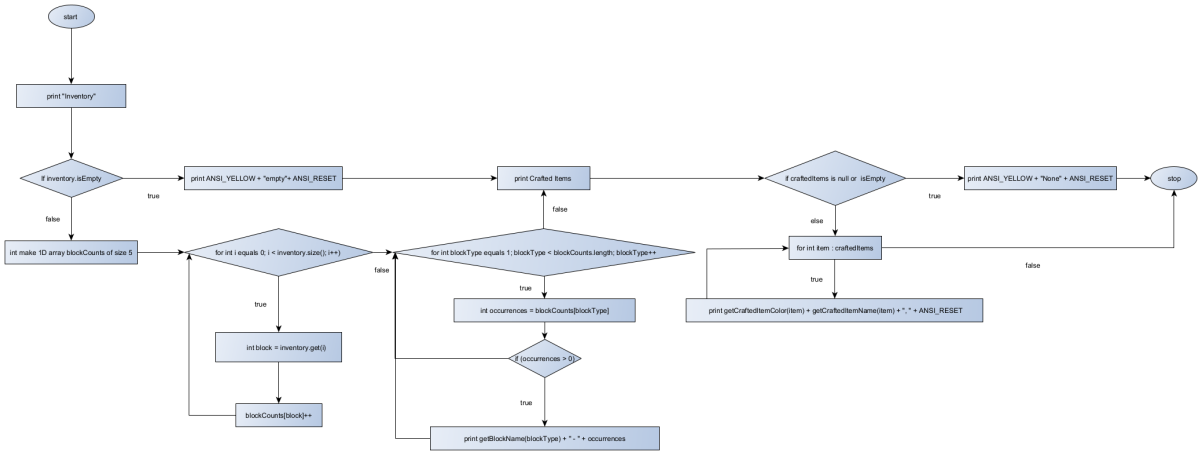



```
function lookAround()
Set playerX be x position of player
Set playerY be y position of player

print("You look around and see:")
FOR y = Math.max(0, playerY - 1); y <= Math.min(playerY + 1, worldHeight - 1); y++:
  FOR x = Math.max(0, playerX - 1); x <= Math.min(playerX + 1, worldWidth - 1); x++:
    if x == playerX and y == playerY:
      print("P");
    else:
      print(block at position [x][y])
  print empty line
print empty line
end function
```

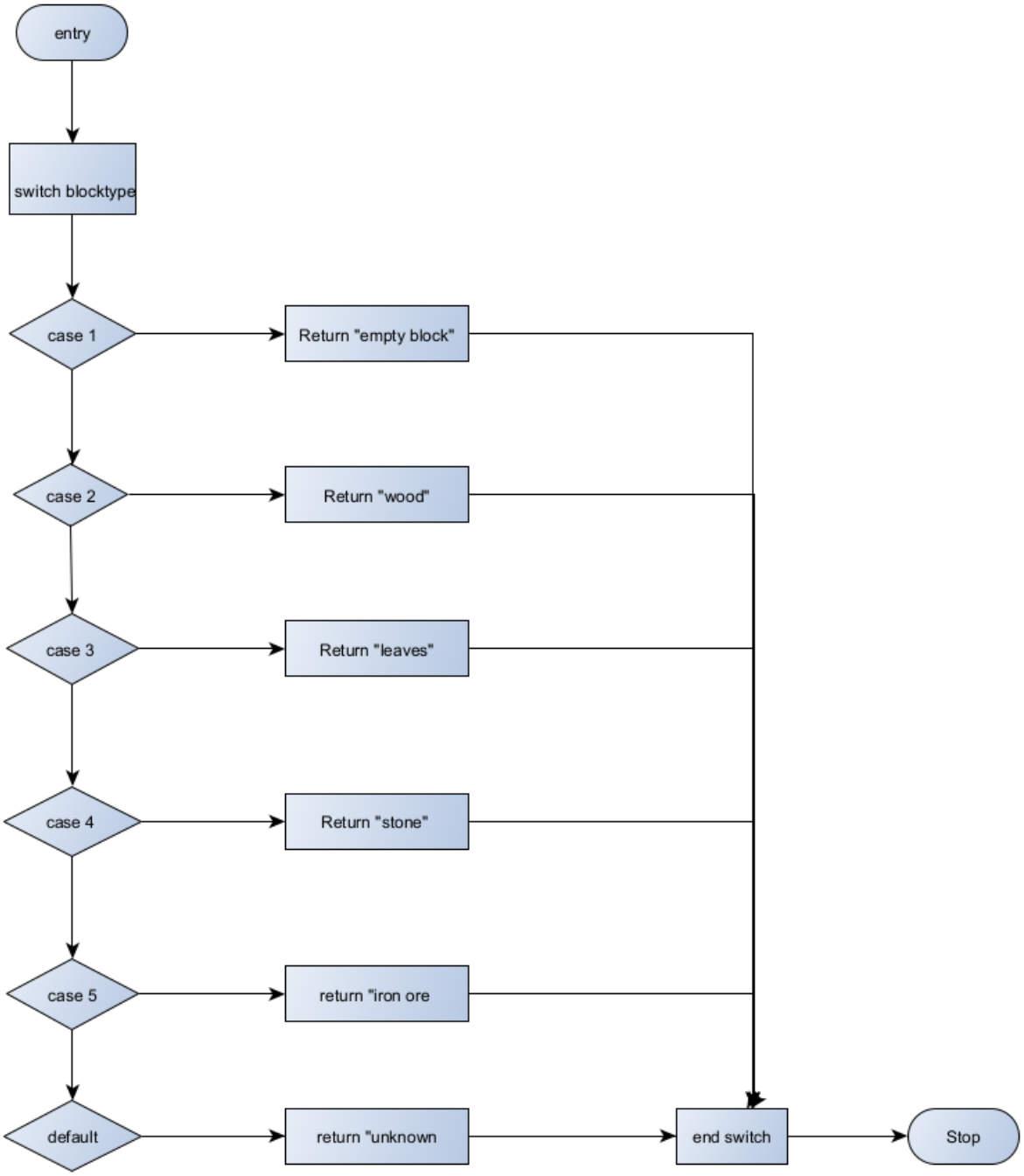



```
function getCountryAndQuoteFromServer():
TRY:
Set link = "https://flag.ashish.nl/get_flag"
Setup a connection to link
Set request method of connection to "POST"
Set request property of connection to "Content-Type" as json
Enable output of connection

let payload be stringified json
let writer be OutputStreamWriter of connection
     Write payload to writer
     Flush writer
     Close writer

     let reader be BufferedReader of connection
     let sb be StringBuilder
     let line be empty string
     WHILE (line is not null):
     let line read next line of reader
     Append line line to sb
         json = ConvertToString(sb)

         let countryStart = FindSubstringIndex(json, " ") + 11
         let countryEnd = FindSubstringIndex(json, " ", countryStart)
         let country = Substring(json, countryStart, countryEnd)

     let quoteStart = FindSubstringIndex(json, " ") + 9
     let quoteEnd = FindSubstringIndex(json, " ", quoteStart)
     let quote = Substring(json, quoteStart, quoteEnd)

     quote = ReplaceSpaces(quote)
     Print("Country: " + country)
     Print("Quote: " + quote)
     CATCH Exception AS e:
```

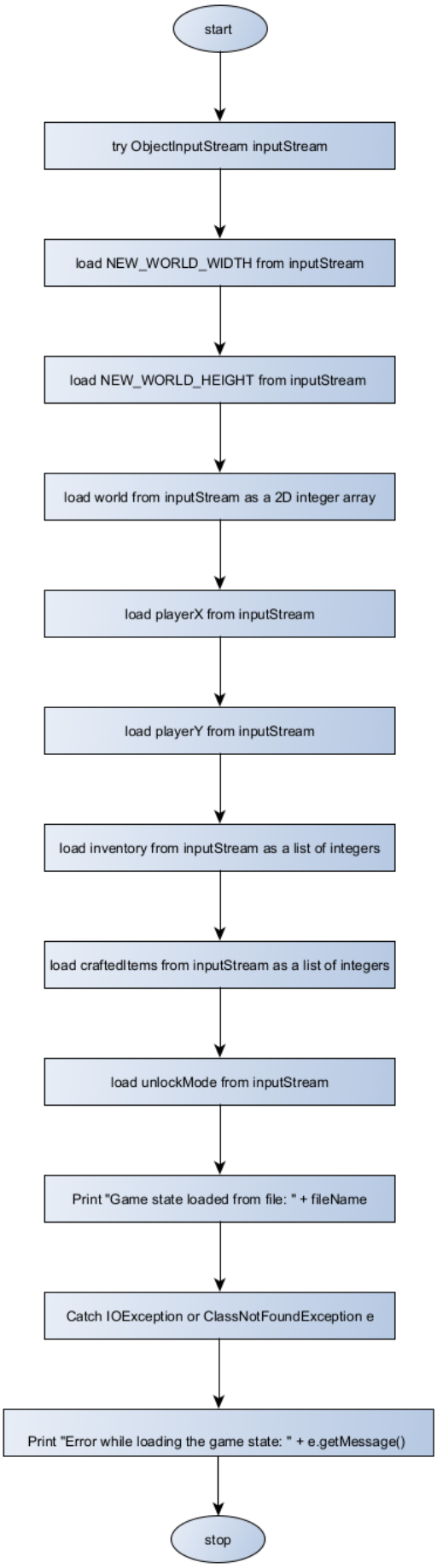
```
        stackTrace = GetStackTrace(e)
        Print("Error connecting to the server")
        Print(stackTrace)
        end function
```



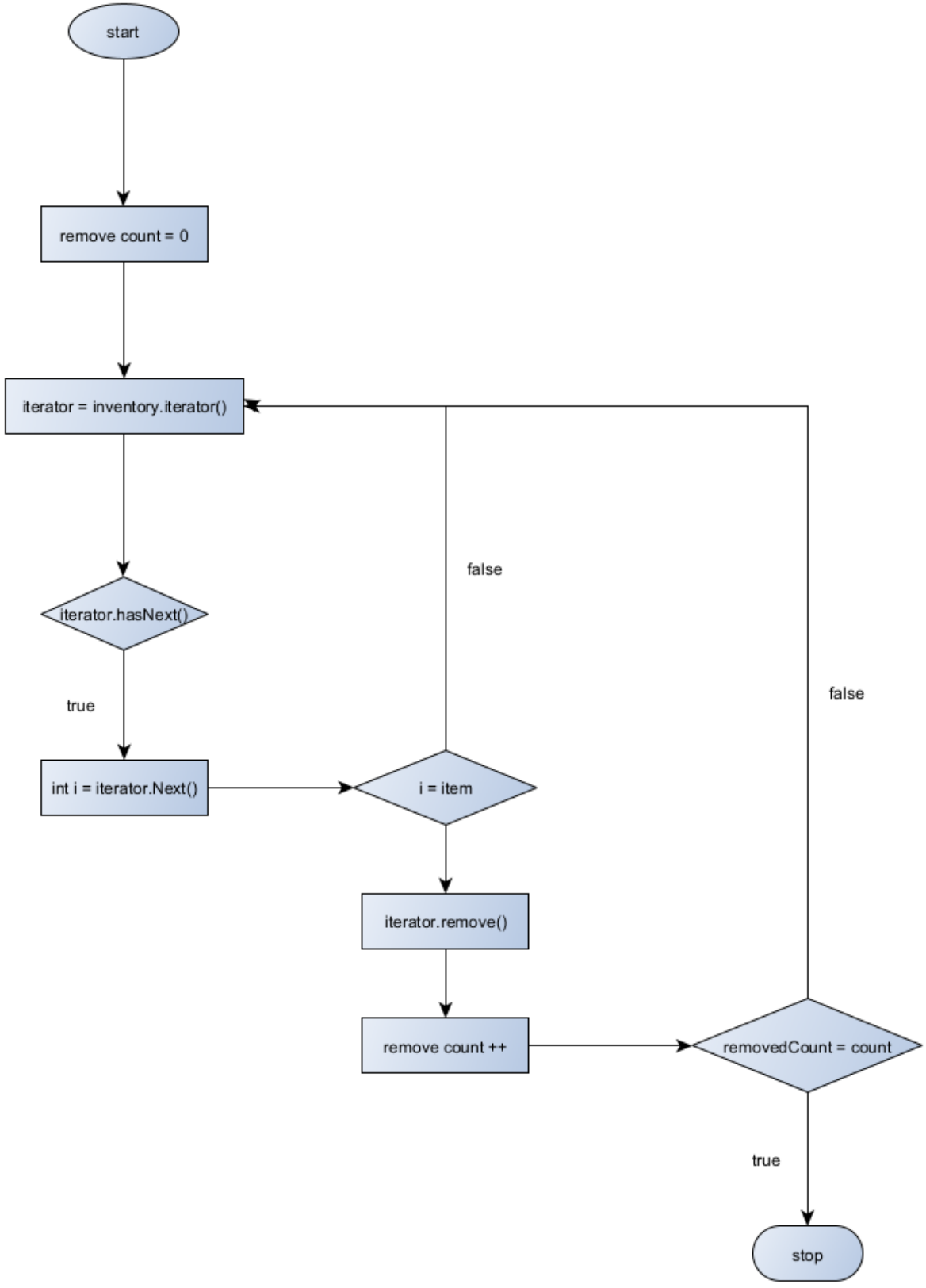

```
  function displayInventory:

 print "Inventory"
 if inventory.isEmpty
    print ANSI_YELLOW + " empty" + ANSI_RESET

else:
   make a 1D array blockCounts of size 5
   for int i = 0; i < inventory.size(); i++
   int block = inventory.get(i)
   blockCounts[block]

For int blockType = 1; blockType < blockCounts.length; blockType++
   int occurrences = blockCounts[blockType]
If occurrences > 0:
   Print getBlockName(blockType) + " - " + occurrences
print "Crafted Item"
if craftedItems is null or craftedItems is empty
   print ANSI_YELLOW_ + "none" + ANSi_RESET
else
   for int item : crafted item
   print getCraftedItemColor(item) + getCraftedItemName(item) + ", " + ANSI_RESET
```




```
Function getBlockName(blockType):

Switch blockType
Case AIR:
 Return "Empty Block"
Case WOOD:
Return "Wood"
Case LEAVES:
Return "Leaves"
Case STONE:
Return "Stone"
 Case IRON_ORE:
Return "Iron Ore"
Default:
 Return "Unknown"
```

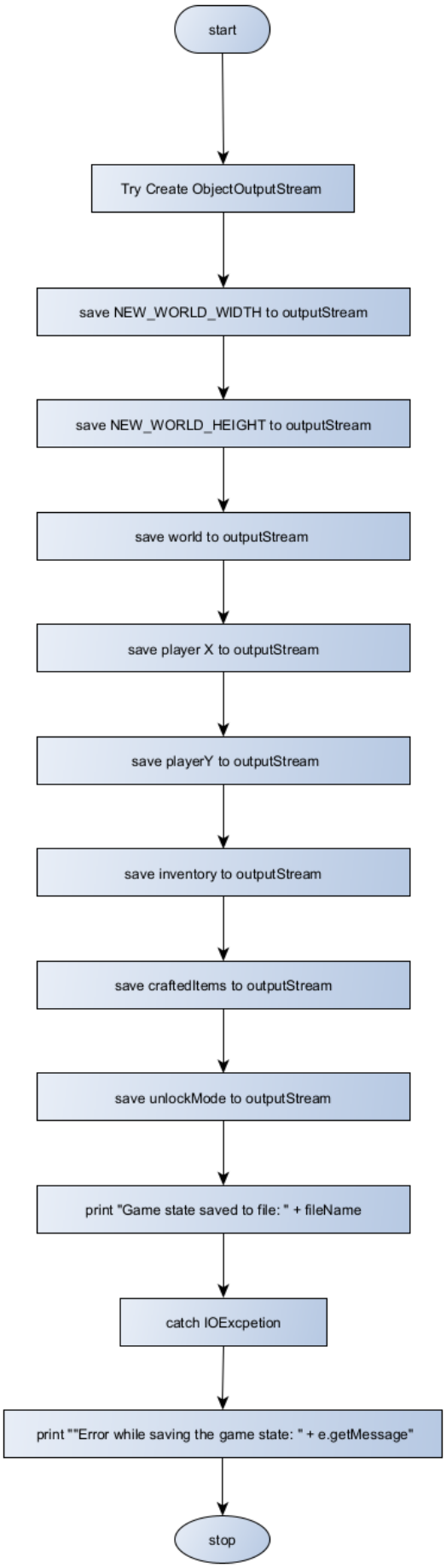

start
try ObjectInputStream inputStream
load NEW_WORLD_WIDTH from inputStream
load NEW_WORLD_HEIGHT from inputStream
load world from inputStream as a 2D integer array
load playerX from inputStream
load playerY from inputStream
load inventory from inputStream as a list of integers
load craftedItems from inputStream as a list of integers
load unlockMode from inputStream
Print "Game state loaded from file: " + fileName
Catch IOException or ClassNotFoundException e
Print "Error while loading the game state: " + e.getMessage()
stop

```
  Function loadGame(fileName)
Try Create ObjectInputStream inputStream using FileInputStream(fileName)
load NEW_WORLD_WIDTH from inputStream
load NEW_WORLD_HEIGHT from inputStream
load world from inputStream as a 2D integer array
load playerX from inputStream
load playerY from inputStream
load inventory from inputStream as a list of integers
load craftedItems from inputStream as a list of integers
load unlockMode from inputStream
Print "Game state loaded from file: " + fileName
 Catch IOException or ClassNotFoundException e
Print "Error while loading the game state: " + e.getMessage()
```




```
function removeItemsFromInventory(item, count):

removedCount = 0
iterator = createIterator(inventory)
While iterator.hasNext()
    i = iterator.next()
```

```
If i is equal to item iterator.remove()
   removedCount = removedCount + 1
If removedCount is equal to count
    break
```


start
Try Create ObjectOutputStream
save NEW_WORLD_WIDTH to outputStream
save NEW_WORLD_HEIGHT to outputStream
save world to outputStream
save player X to outputStream
save playerY to outputStream
save inventory to outputStream
save craftedItems to outputStream
save unlockMode to outputStream
print "Game state saved to file: " + fileName
catch IOExcpetion
print ""Error while saving the game state: " + e.getMessage"
stop

```
  function saveGame(fileName):

Try  ObjectOutputStream outputStream
 save NEW_WORLD_WIDTH to outputStream
save NEW_WORLD_HEIGHT to outputStream
 save world to outputStream
save playerX to outputStream
 save playerY to outputStream
 save inventory to outputStream
save craftedItems to outputStream
save unlockMode to outputStream
 Print "Game state saved to file: " + fileName Catch IOException e
Print "Error while saving the game state: " + e.getMessage()
```