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) + "╝"

Function **getBlockSymbol** with parameter '**blockType**':

* Declare 'blockColor'
* Switch 'blockType':
* Case AIR:

Return "- "

* Case WOOD:

Set 'blockColor' to ANSI\_RED

* Case LEAVES:

Set 'blockColor' to ANSI\_GREEN

* Case STONE:

Set 'blockColor' to ANSI\_BLUE

* Case IRON\_ORE:

Set 'blockColor' to ANSI\_WHITE

* Case GOLD\_ORE:

Set 'blockColor' to ANSI\_YELLOW

* Case DIAMOND\_ORE:

Set 'blockColor' to ANSI\_CYAN

* Default:

Set 'blockColor' to ANSI\_RESET

* Return 'blockColor' + getBlockChar(blockType) + " "

Function **getBlockChar** with parameter '**blockType**':

* Switch 'blockType':
* Case WOOD:

Return '\u2592'

* Case LEAVES:

Return '\u00A7'

* Case STONE:

Return '\u2593'

* Case IRON\_ORE:

Return '\u00B0'

* Case GOLD\_ORE:

Return '\u0E51'

* Case DIAMOND\_ORE:

Return '\u09F0'

* Default:

Return '-'

Function **startGame**:

* Initialize scanner object 'scanner' for user input
* Initialize 'unlockMode', 'craftingCommandEntered', 'miningCommandEntered', 'movementCommandEntered', 'openCommandEntered' to false
* While true:

Clear the screen

Print legend

Print world

Print inventory

Print action instructions

* Read user input into 'input' and convert it to lowercase
* If 'input' is one of ["w", "up", "s", "down", "a", "left", "d", "right"]:
* If 'unlockMode':

Set 'movementCommandEntered' to true

Move player based on 'input'

* Else if 'input' is "m":

If 'unlockMode':

Set 'miningCommandEntered' to true

Mine block

* Else if 'input' is "p":

Print inventory

Prompt user for 'blockType' to place

Place block based on 'blockType'

* Else if 'input' is "c":

Print crafting recipes

Prompt user for 'recipe' to craft

Craft item based on 'recipe'

* Else if 'input' is "i":

Interact with the world

* Else if 'input' is "save":

Prompt user for 'fileName' to save the game state

Save game state with 'fileName'

* Else if 'input' is "load":

Prompt user for 'fileName' to load the game state

Load game state from 'fileName'

* Else if 'input' is "exit":

Print "Exiting the game. Goodbye!"

Break the loop

* Else if 'input' is "look":

Look around in the world

* Else if 'input' is "unlock":

Set 'unlockMode' to true

* Else if 'input' is "getflag":

Get country and quote from server

Wait for user to press Enter

* Else if 'input' is "open":

If 'unlockMode' and 'craftingCommandEntered' and 'miningCommandEntered' and 'movementCommandEntered':

Set 'secretDoorUnlocked' to true

Reset the world

Print "Secret door unlocked!"

Wait for user to press Enter

* Else:

Print "Invalid passkey. Try again!"

Wait for user to press Enter

Reset flags

* Else:

Print "Invalid input. Please try again."

* If 'unlockMode':
* If 'input' is "c":

Set 'craftingCommandEntered' to true

* Else if 'input' is "m":

Set 'miningCommandEntered' to true

* Else if 'input' is "open":

Set 'openCommandEntered' to true

* If 'secretDoorUnlocked':

Clear the screen

Print "You have entered the secret area!"

Print "You are now presented with a game board with a flag!"

Set 'inSecretArea' to true

Reset the world

Reset 'secretDoorUnlocked'

Fill the inventory

Wait for user to press Enter

Function **fillInventory**:

* Clear the inventory
* For 'blockType' from 1 to 6:
* For 'i' from 0 to INVENTORY\_SIZE:

Add 'blockType' to the inventory

Function **resetWorld**:

* Call generateEmptyWorld
* Set 'playerX' to NEW\_WORLD\_WIDTH / 2
* Set 'playerY' to NEW\_WORLD\_HEIGHT / 2

Function **generateEmptyWorld**:

* Create a new world with dimensions NEW\_WORLD\_WIDTH x NEW\_WORLD\_HEIGHT
* Set 'redBlock' to 1
* Set 'whiteBlock' to 4
* Set 'blueBlock' to 3
* Set 'stripeHeight' to NEW\_WORLD\_HEIGHT / 3
* To fill the top stripe with red blocks

For 'y' from 0 to 'stripeHeight':

For 'x' from 0 to NEW\_WORLD\_WIDTH:

Set world[x][y] to 'redBlock'

* To fill the middle stripe with white blocks

For 'y' from 'stripeHeight' to 'stripeHeight' \* 2:

For 'x' from 0 to NEW\_WORLD\_WIDTH:

Set world[x][y] to 'whiteBlock'

* To fill the bottom stripe with blue blocks

For 'y' from 'stripeHeight' \* 2 to NEW\_WORLD\_HEIGHT:

For 'x' from 0 to NEW\_WORLD\_WIDTH:

Set world[x][y] to 'blueBlock'

Function **clearScreen**:

* Try:

If OS is Windows:

Execute "cmd /c cls" to clear the screen

* Else:

Execute ANSI escape codes to clear the screen

Catch and print any exceptions

Function **lookAround**:

* Print "You look around and see:"
* For 'y' from (playerY - 1) to (playerY + 1):
* For 'x' from (playerX - 1) to (playerX + 1):
* If 'x' equals 'playerX' and 'y' equals 'playerY':

Print ANSI\_GREEN + "P " + ANSI\_RESET

* Else:

Print getBlockSymbol(world[x][y])

Print a newline

Print a newline

WaitForEnter()

Function **movePlayer** with parameter '**direction**':

* Convert 'direction' to uppercase
* Switch 'direction':
* Case "W", "UP":
* If playerY > 0:

Decrement playerY

* Case "S", "DOWN":
* If playerY < (worldHeight - 1):

Increment playerY

* Case "A", "LEFT":
* If playerX > 0:

Decrement playerX

* Case "D", "RIGHT":
* If playerX < (worldWidth - 1):

Increment playerX

Function **mineBlock**:

* Get 'blockType' from world[playerX][playerY]
* If 'blockType' is not AIR:
* If ('blockType' is DIAMOND\_ORE or 'blockType' is GOLD\_ORE) and not craftedItemsContains(CRAFTED\_IRON\_PICKAXE, 1):

Print "Can not mine this"

* Else:

Add 'blockType' to inventory

Set world[playerX][playerY] to AIR

Print "Mined " + getBlockName(blockType)

* Else:

Print "No block to mine here."

WaitForEnter()

Function **placeBlock** with parameter '**blockType**':

* If 'blockType' is between 0 and 9:
* If 'blockType' is less than or equal to 6:
* If inventory contains 'blockType':

Remove 'blockType' from inventory

Set world[playerX][playerY] to 'blockType'

Print "Placed " + getBlockName(blockType) + " at your position."

* Else:

Print "You don't have " + getBlockName(blockType) + " in your inventory."

* Else:

Get 'craftedItem' from 'blockType'

If craftedItems contains 'craftedItem':

Remove 'craftedItem' from craftedItems

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."

* Else:

Print "Invalid block number. Please enter a valid block number."

Print BLOCK\_NUMBERS\_INFO

WaitForEnter()

Function **getBlockTypeFromCraftedItem** with parameter '**craftedItem**':

* Switch 'craftedItem':
* Case CRAFTED\_WOODEN\_PLANKS:

Return 5

* Case CRAFTED\_STICK:

Return 6

* Case CRAFTED\_IRON\_INGOT:

Return 7

* Case CRAFTED\_IRON\_PICKAXE:

Return 8

* Case CRAFTED\_GOLD\_INGOT:

Return 9

* Case CRAFTED\_DIAMOND\_INGOT:

Return 10

* Default:

Return -1

Function **getCraftedItemFromBlockType** with parameter '**blockType**':

* Switch 'blockType':
* Case 5:

Return CRAFTED\_WOODEN\_PLANKS

* Case 6:

Return CRAFTED\_STICK

* Case 7:

Return CRAFTED\_IRON\_INGOT

* Case 8:

Return CRAFTED\_IRON\_PICKAXE

* Case 9:

Return CRAFTED\_GOLD\_INGOT

* Case 10:

Return CRAFTED\_DIAMOND\_INGOT

* Default:

Return -1

Function **PrintCraftingRecipes**:

* 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"

Function **craftItem** with parameter '**recipe**':

* Switch 'recipe':
* Case 1:

Call craftWoodenPlanks

* Case 2:

Call craftStick

* Case 3:

Call craftIronIngot

* Case 4:

Call craftIronPickaxe

* Case 5:

Call craftGoldIngot

* Case 6:

Call craftDiamondIngot

* Default:

Print "Invalid recipe number."

WaitForEnter()

Function **craftWoodenPlanks**:

* If inventoryContains(WOOD, 2):

RemoveItemsFromInventory(WOOD, 2)

AddCraftedItem(CRAFTED\_WOODEN\_PLANKS)

Print "Crafted Wooden Planks."

* Else:

Print "Insufficient resources to craft Wooden Planks."

Function **craftStick**:

* If inventoryContains(WOOD):

RemoveItemsFromInventory(WOOD, 1)

AddCraftedItem(CRAFTED\_STICK)

Print "Crafted Stick."

* Else:

Print "Insufficient resources to craft Stick."

Function **craftIronIngot**:

* If inventoryContains(IRON\_ORE, 3):

RemoveItemsFromInventory(IRON\_ORE, 3)

AddCraftedItem(CRAFTED\_IRON\_INGOT)

Print "Crafted Iron Ingot."

* Else:

Print "Insufficient resources to craft Iron Ingot."

Function **craftIronPickaxe**:

* If inventoryContains(IRON\_ORE, 3) and craftedItemsContains(CRAFTED\_STICK, 2):

RemoveItemsFromInventory(IRON\_ORE, 3)

RemoveItemsFromCraftedItems(CRAFTED\_STICK, 2)

AddCraftedItem(CRAFTED\_IRON\_PICKAXE)

Print "Crafted Iron Pickaxe."

* Else:

Print "Insufficient resources to craft Iron Pickaxe."

Function **craftGoldIngot**:

* If inventoryContains(GOLD\_ORE, 1):

RemoveItemsFromInventory(GOLD\_ORE, 1)

AddCraftedItem(CRAFTED\_GOLD\_INGOT)

Print "Crafted Gold Ingot."

* Else:

Print "Insufficient resources to craft Gold Ingot."

Function **craftDiamondIngot**:

* If inventoryContains(DIAMOND\_ORE, 1):

RemoveItemsFromInventory(DIAMOND\_ORE, 1)

AddCraftedItem(CRAFTED\_DIAMOND\_INGOT)

Print "Crafted Diamond Ingot."

* Else:

Print "Insufficient resources to craft Diamond Ingot."

Function **inventoryContains** with parameter '**item**':

* Return true if inventory contains 'item', otherwise return false.

Function **inventoryContains** with parameters '**item**' and '**count**':

* Initialize 'itemCount' to 0
* For each 'i' in inventory:

If 'i' equals 'item':

Increment 'itemCount'

If 'itemCount' equals 'count':

Return true

Return false

Function **removeItemsFromInventory** with parameters '**item**' and '**count**':

* Initialize 'removedCount' to 0
* Initialize an iterator for inventory
* While iterator has next element:

Get 'i' from iterator

If 'i' equals 'item':

Remove 'i' from iterator

Increment 'removedCount'

If 'removedCount' equals 'count':

Break the loop

Function **addCraftedItem** with parameter '**craftedItem**':

* If craftedItems is null:

Initialize craftedItems as an empty list

* Add 'craftedItem' to craftedItems

Function **craftedItemsContains** with parameters '**item**' and '**count**':

* Initialize 'itemCount' to 0
* For each 'i' in craftedItems:

If 'i' equals 'item':

Increment 'itemCount'

If 'itemCount' equals 'count':

Return true

Return false

Function **removeItemsFromCraftedItems** with parameters '**item**' and '**count**':

* Initialize 'removedCount' to 0
* Initialize an iterator for craftedItems
* While iterator has next element:

Get 'i' from iterator

* If 'i' equals 'item':

Remove 'i' from iterator

* Increment 'removedCount'
* If 'removedCount' equals 'count':

Break the loop

Function **interactWithWorld**:

* Get 'blockType' from world[playerX][playerY]
* Switch 'blockType':
* Case WOOD:

Print "You gather wood from the tree."

Add WOOD to inventory

* Case LEAVES:

Print "You gather leaves from the tree."

Add LEAVES to inventory

* Case STONE:

Print "You gather stones from the ground."

Add STONE to inventory

* Case IRON\_ORE:

Print "You mine iron ore from the ground."

Add IRON\_ORE to inventory

* Case DIAMOND\_ORE:

If inventoryContains(CRAFTED\_IRON\_PICKAXE, 1):

Print "You mine diamond ore from the ground."

Add DIAMOND\_ORE to inventory

Else:

Print "You can not mine this!"

* Case GOLD\_ORE:

If inventoryContains(CRAFTED\_IRON\_PICKAXE, 1):

Print "You mine gold ore from the ground."

Add GOLD\_ORE to inventory

Else:

Print "You can not mine this!"

* Case AIR:

Print "Nothing to interact with here."

* Default:

Print "Unrecognized block. Cannot interact."

* WaitForEnter()

Function **saveGame** with parameter '**fileName**':

* Try:
* Create an ObjectOutputStream with FileOutputStream for 'fileName'
* Write game state data to the file

Print "Game state saved to file: " + fileName

* Catch IOException 'e':

Print "Error while saving the game state: " + e.getMessage()

WaitForEnter()

Function **loadGame** with parameter '**fileName**':

* Try:
* Create an ObjectInputStream with FileInputStream for 'fileName'
* Read game state data from the file and load it into the program

Print "Game state loaded from file: " + fileName

* Catch IOException or ClassNotFoundException 'e':

Print "Error while loading the game state: " + e.getMessage()

WaitForEnter()