Criterion C: Development

Techniques:

1. String array
2. Different methods used in the program. Methods are defined in Game.java and created by me
3. Multiple commands are used for different outcomes in the code
4. Interconnected methods
5. Use of scanners

String Array:

```
String inventory[] = new String[20];
```

This is the initialization of the String Array. It can store any 20 strings and is used in my program as the inventory system.

```
} else if (response.equals("inventory")) {

    for (int i = 0; i < 20; i++) {
        if (inventory[i] != null) {
            System.out.println(inventory[i]);
        }
    }
```

This string of code is what runs when the player enters “Inventory”. The code loops through all 20 nodes of the array and outputs to the player the nodes that are populated or ,in game terms, the items that they picked up.

```
System.out.println("You pick up the gun, a standard laser pistol, this might come in handy");
for (int i = 0; i < 20; i++) {
    if (inventory[i] != null) {
    } else {
        inventory[i] = "Standard laser pistol";
        i = 20;
    }
}
```

The string of code above is the code that is used when the player picks up an item in the game. The code loops through the array and checks for an empty slot. When it finds one it enters the item into that node in the form of a string and ends the loop.

This string array proves effective since all the items are stored in one place and not their separate variables.

Different Methods:

```
public static void hackerdeviceminigame(String response, Scanner in) {
    int counter = 1;
    int guess;
    Random myrand = new Random();
    int number;

    number = myrand.nextInt(10000);

    System.out.println(
            "The lock hacker device has narrowed the code needed to open the lock down to a number between 0 and 10000. Enter a number to narrow down the code.");

    guess = in.nextInt();

    while (guess != number) {
        counter++;
        if (guess > number) {
            System.out.println("Too High");
        } else {
            if (guess < number) {
                System.out.println("Too Low");
            }
        }
        guess = in.nextInt();
    }
    System.out.println("You have opened the door and entered the power room.");
    power(response, in);
}
```

This is an example of one of the methods in my program. This method has a very specific purpose and is named according to what it does, for example this method is called hackerdeviceminigame and is what it says, a minigame. This method is specifically a minigame that runs when the player uses the lock hacker device.

Every room in the game has its own method for better organization and readability of the code. Here are some examples of the names of the methods of the rooms.

```
public static void staff(String response, Scanner in) {
```

```
public static void fourwayfrompower(String response, Scanner in) {
```

```
public static void fourwayfromstaff(String response, Scanner in) {
```

```
 public static void fourwayfromcryo(String response, Scanner in) {
```

```
public static void cryo(String response, Scanner in) {
```

```
private static void powerroom(String response, Scanner in) {
```

Use of Scanners:

Each method in my code uses Scanners to log the response of the player. As shown above, every method in the game has Scanner ‘in’ and String ‘response’. The Scanner detects the input of the player and puts it in the String ‘response’. ‘Response’ then determines what the code will do next. Below is an example of the interaction between the scanner and response, in.next() being the scanner.

```
System.out.println("What is your move?");

response = in.next();

if (response.equals("laser") && havegun == true) {
    damage = myrand.nextInt(20);
    alienhp = alienhp - damage;
    System.out.println("You hit alien for " + damage + " damage");

    if (stun == false) {
        damage = myrand.nextInt(20);
        hp = hp - damage;
        System.out.println("Alien hit you for " + damage + " damage");
    } else {
        stun = false;
    }
```

Multiple commands:

Multiple commands are used for the player to interact freely with the environment and the game. Here is an example of multiple commands being used

```
response = in.next();
if (response.equals("enterpassword")) {
    System.out.println("Enter password below");
    response = in.next();
    if (response.equals("crumble")) {
        System.out.println("the door opens and you enter admin");
        admin(response, in);
    } else if (response.equals("quit")) {
        System.out.println("you have quit and returned to the intersection");
    } else {
        System.out.println("Incorrect password, try again, or press quit to quit");
    }
} else if (response.equals("quit")) {
    System.out.println("You have quit and returned to the intersection");
    fourwayfromadmin(response, in);
} else {
    System.out.println("invalid command");
}
```

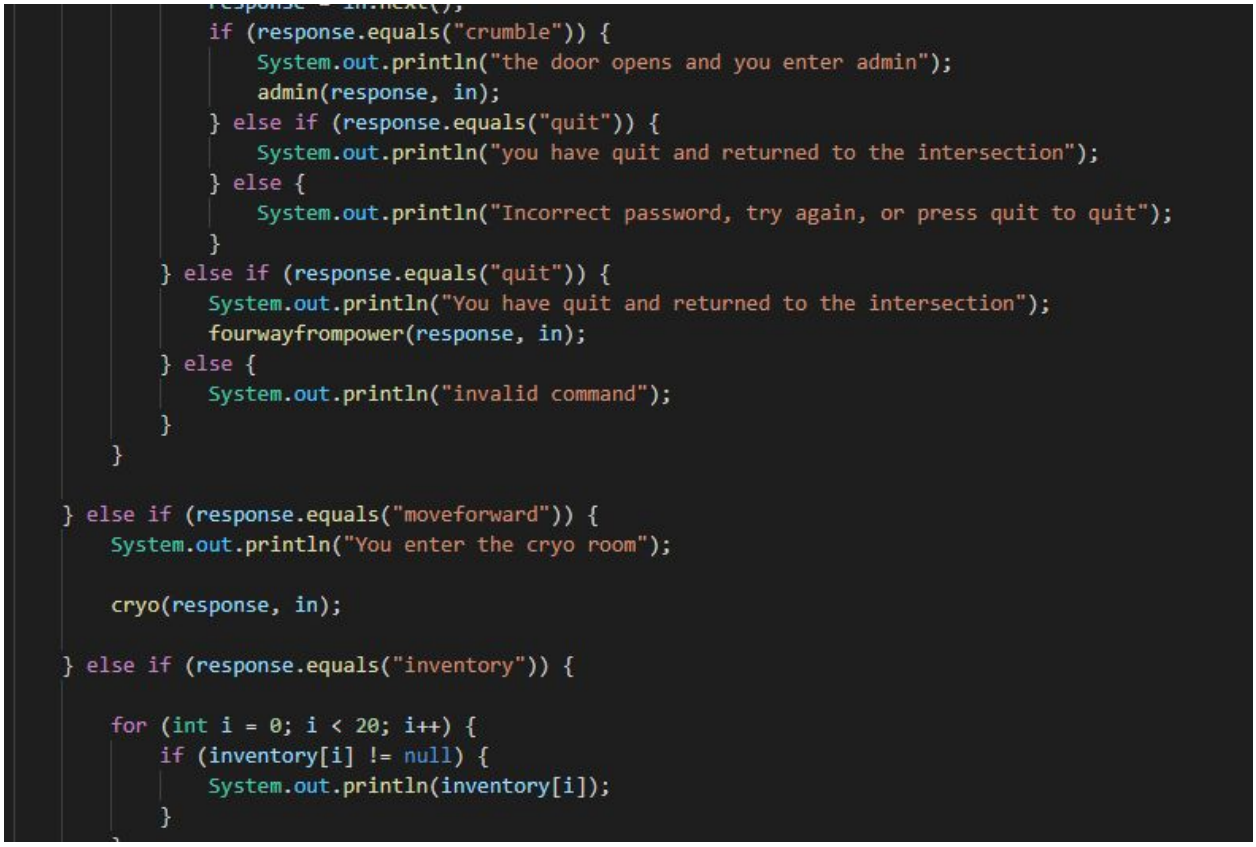
As shown above, the if statements go through all the commands that can be used, checking which one the player put. Each command has a different outcome as shown in the code and elicits a different response by the program. This provides the program with multiple commands and multiple different outcomes. This provides the user with more options that they can choose in the program and providing more outcomes that the program has.

Interconnected Methods:

Every method in the program is connected to many other methods. This generates a web in the code where every method is interconnected and the player can go through each method fluidly. An example of connections between methods are below

```
                    System.out.println("the door opens and you enter admin");
                    admin(response, in);
                } else if (response.equals("quit")) {
                    System.out.println("you have quit and returned to the intersection");
                } else {
                    System.out.println("Incorrect password, try again, or press quit to quit");
                }
            } else if (response.equals("quit")) {
                System.out.println("You have quit and returned to the intersection");
                fourwayfromcryo(response, in);
            } else {
                System.out.println("invalid command");
            }
        }

    } else if (response.equals("movebackward")) {
        System.out.println("You enter the Cryo room");
        cryo(response, in);

    } else if (response.equals("moveright")) {
        System.out.println("You enter the staff quarters");
        staff(response, in);
```

```
                "The power room door is sealed, the lock is sophisticated and you cant open it.");
        } else {
            System.out.println("You place the hacker device over the lock");
            hackerdeviceminigame(response, in);
        }
    } else {
        System.out.println("You enter the power room");
        power(response, in);
    }

} else if (response.equals("moveforward")) {
    System.out.println(
            "The admin office door is locked, it needs some sort of password. (type enterpassword to type the password or quit to quit)");

    while (response != "quit") {
        response = in.next();
        if (response.equals("enterpassword")) {
            System.out.println("Enter password below");
            response = in.next();
            if (response.equals("crumble")) {
                System.out.println("the door opens and you enter admin");
                admin(response, in);
            } else if (response.equals("quit")) {
                System.out.println("you have quit and returned to the intersection");
            } else {
                System.out.println("Incorrect password, try again, or press quit to quit");
            }
        } else if (response.equals("quit")) {
            System.out.println("You have quit and returned to the intersection");
            fourwayfromstaff(response, in);
        } else {
```



```
            if (response.equals("crumble")) {
                System.out.println("the door opens and you enter admin");
                admin(response, in);
            } else if (response.equals("quit")) {
                System.out.println("you have quit and returned to the intersection");
            } else {
                System.out.println("Incorrect password, try again, or press quit to quit");
            }
        } else if (response.equals("quit")) {
            System.out.println("You have quit and returned to the intersection");
            fourwayfrompower(response, in);
        } else {
            System.out.println("invalid command");
        }
    }

} else if (response.equals("moveforward")) {
    System.out.println("You enter the cryo room");

    cryo(response, in);

} else if (response.equals("inventory")) {

    for (int i = 0; i < 20; i++) {
        if (inventory[i] != null) {
            System.out.println(inventory[i]);
        }
```

Each of these methods above have several calls to other methods. Each picture shows at least 3 calls. Each method contains these calls to other methods, and those calls connect every method with the other in a web of possibilities of going through methods. This provides the user of the program to easily transfer between methods and for the program to be well organized and labeled with methods while not sacrificing the fluidity and functionality of the product.

Word count: 562