BACS2063 Data Structures and Algorithms

ASSIGNMENT 202005

Student Name	: **TEO WEN ZHI**
Student ID	: **20WMR09522**
Programme	: **BACHELOR OF COMPUTER SCIENCE (HONOURS) IN SOFTWARE ENGINEERING**
Tutorial Group	: **RSF Group 7**
Team Name	: **BigBrain**

Declaration

- I confirm that I have read and complied with all the terms and conditions of Tunku Abdul Rahman University College's plagiarism policy.
- I declare that this assignment is free from all forms of plagiarism and for all intents and purposes is my own properly derived work.

[illegible]

Student's signature

23-8-2020

✓

Date

Table of Contents

✓

1. Introduction to the Game

BigBrain is a kind of quiz board game, the concept is players need to answer a set of questions and for each of the questions and they have 3 life in a round of game. When they answer correctly will award some score for them, if answer wrongly will deduct the life of player, at the end of game the player who gets the highest score and alive will be the winner. To increase the diversification and fun of this game, we came up with some ideas for BigBrain which is **use of skill** and **animal achievement**. For the skill idea, we provide each player a chance to choose 1 skill for their game journey, the skill might be very useful or useless depending on the luck of the player. Then for the animal achievement is a kind of evaluate mechanism which depends on the total score of the player award at the end of the game, providing an animal title for them. It also will be the achievement for the player which records the times of the player achieving the specific animal title.

2. User Guide

Overall guide

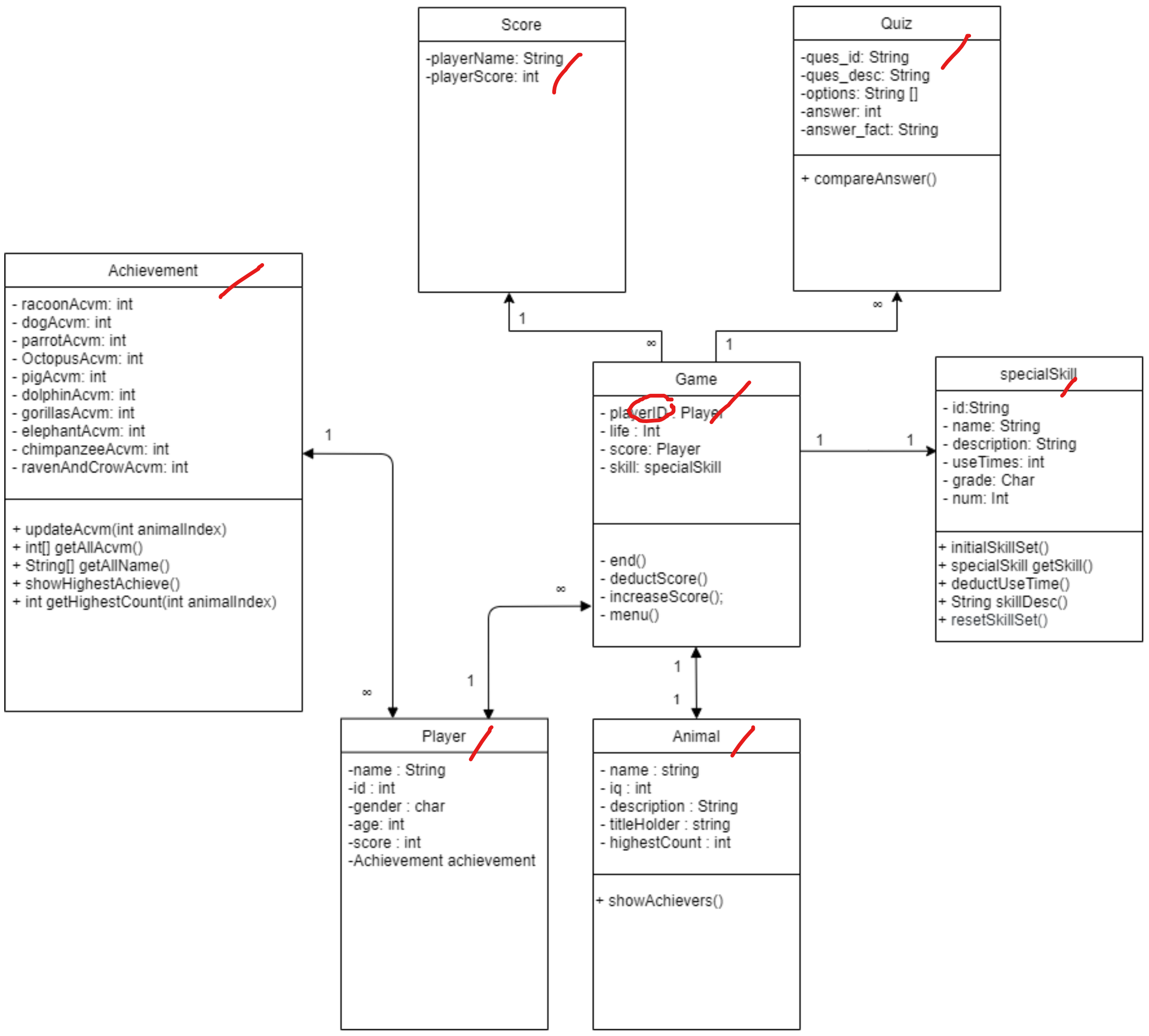
- When user starts the game, they should able to see 5 option which is Start Game, achievement, scoreboard and quiz.
- Choose the Start game will allow user, select the player from default game player list, or they can add the new player or delete the player in the player list
- Choose the achievement will show all the animal title holder.
- Choose the scoreboard will let user display the top 10 players or reset the scoreboard.
- Choose on the quiz will allow user do search, add, edit and remove for the quiz in this game.

X/o UI ?

Gameplay guide

- Choose the start game at the main menu.
- You will need to select the game mode you prefer from 1 to 4 players game mode.
- Now you need select the player which will represent you in the game from player list.
- After you selected the player will get into game you will need select a skill for them
- They are 2 unknown skill you need to choose one of them as your game skill.
- After you choose the skill that will show some information for your chosen skill.
- Now the quiz will start, number of 1,2,3,4 will represent the option for you answer the question, and number 9 will be key for you to use your game skill.
- When you press 9 in your turn, it will show your skill and you are able to use if that still available
- Some skill is automatic use and some skill is need you to manual use it.
- BigBrain included different type of skills some skills will help you in game journey some might not.
- When all the question answered or all of the run out of life, the game will end and show the gameplay summary for you and your friends.

3. Entity Class Diagram

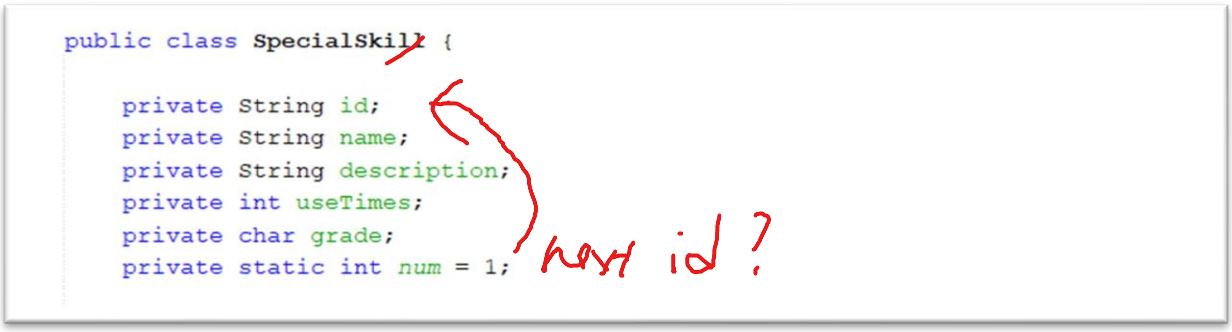


4. Implementation of Entity Class(es)

The entity class I mainly in charge for the BigBrain quiz game is the **SpecialSkill, SkillFunction**, and **Game.**

The **SpecialSkill** is a class will represent the skill will use in the BigBrain.

SpecialSkill data field- all of the data field is describe the characteristics except static num is for generate id.

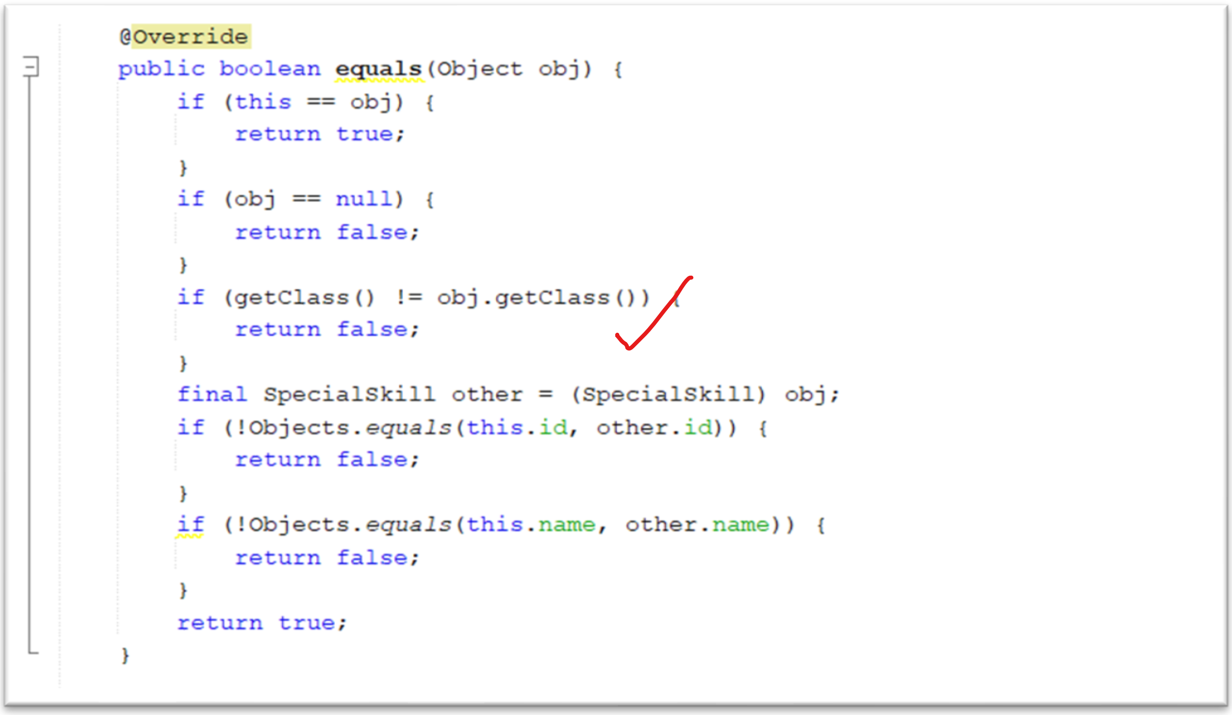


```java
public class SpecialSkill {

    private String id;
    private String name;
    private String description;
    private int useTimes;
    private char grade;
    private static int num = 1;
```

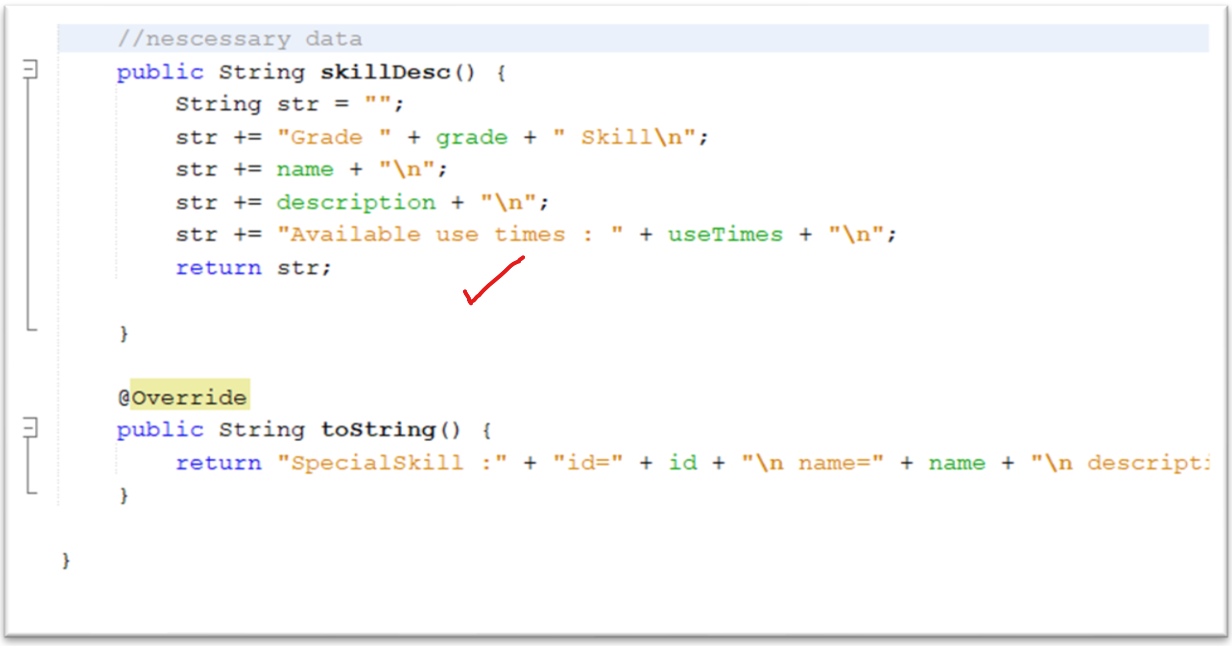
next id ?

SpecialSkill function

Function equals is do comparing base on name and Id skill if they are same will return true. Means the they are 2 same skill.



```java
@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    final SpecialSkill other = (SpecialSkill) obj;
    if (!Objects.equals(this.id, other.id)) {
        return false;
    }
    if (!Objects.equals(this.name, other.name)) {
        return false;
    }
    return true;
}
```

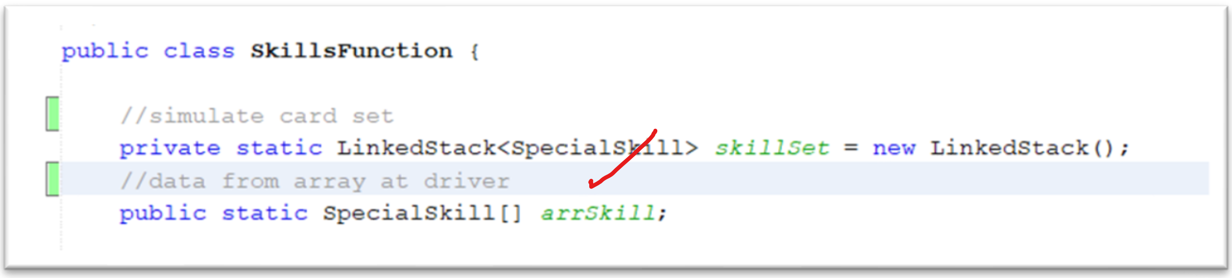


```
    //nescessary data
    public String skillDesc() {
        String str = "";
        str += "Grade " + grade + " Skill\n";
        str += name + "\n";
        str += description + "\n";
        str += "Available use times : " + useTimes + "\n";
        return str;
                              ✓
    }

    @Override
    public String toString() {
        return "SpecialSkill :" + "id=" + id + "\n name=" + name + "\n descript
    }

}
```

The **SkillFunction** class is the object will be doing the relevant skill control to the BigBrain

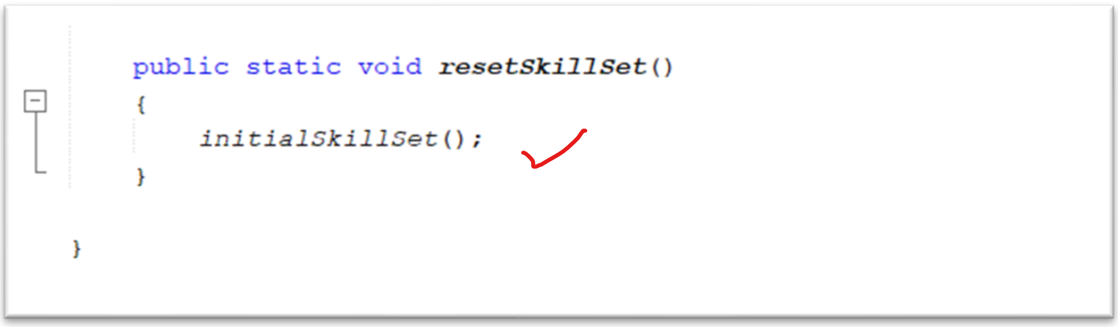
SkillFunction data field

- skillSet is the stack will store the SpecialSkill
- arrSkill is the array for store hard code data.

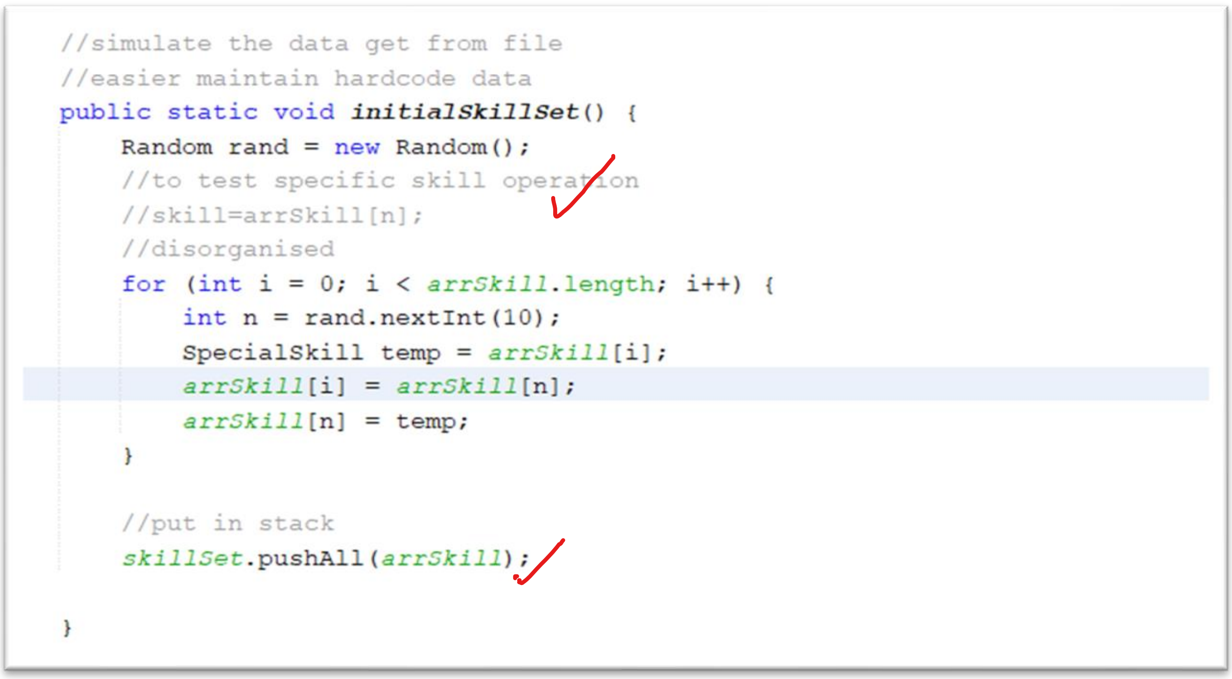


```
public class SkillsFunction {

    //simulate card set
    private static LinkedStack<SpecialSkill> skillSet = new LinkedStack();
    //data from array at driver
    public static SpecialSkill[] arrSkill;
```

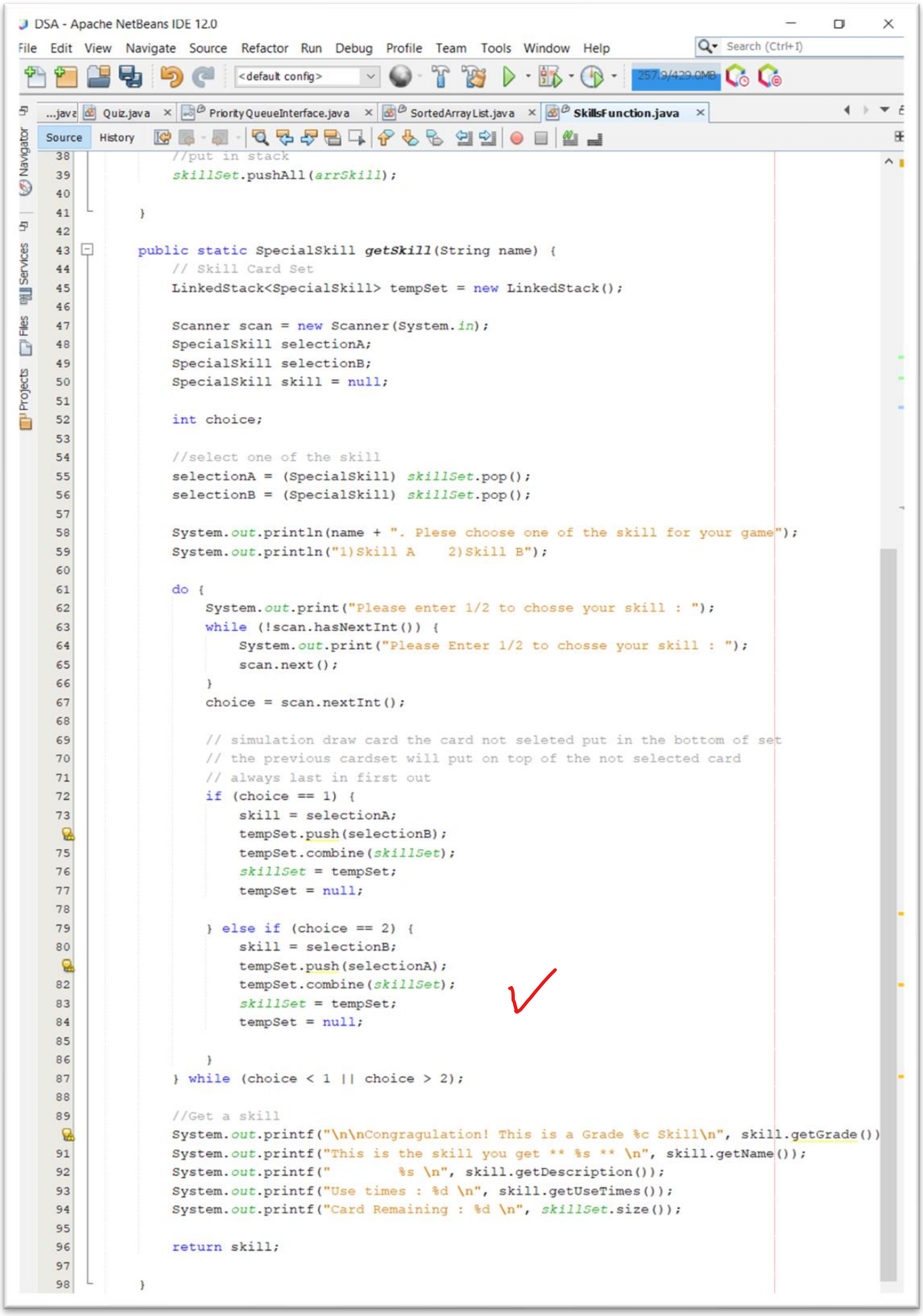
SkillFunction function



```
    public static void resetSkillSet()
    {
        initialSkillSet();
    }

}
```

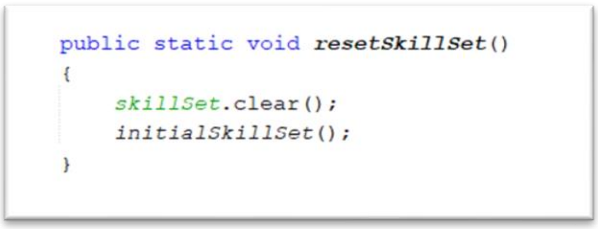


```
//simulate the data get from file
//easier maintain hardcode data
public static void initialSkillSet() {
    Random rand = new Random();
    //to test specific skill operation
    //skill=arrSkill[n];
    //disorganised
    for (int i = 0; i < arrSkill.length; i++) {
        int n = rand.nextInt(10);
        SpecialSkill temp = arrSkill[i];
        arrSkill[i] = arrSkill[n];
        arrSkill[n] = temp;
    }

    //put in stack
    skillSet.pushAll(arrSkill);

}
```


DSA - Apache NetBeans IDE 12.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
257.9/429.0MB
...java Quiz.java PriorityQueueInterface.java SortedArrayList.java SkillsFunction.java
Source History
Navigator Services Files Projects

```java
        //put in stack
        skillSet.pushAll(arrSkill);

    }

    public static SpecialSkill getSkill(String name) {
        // Skill Card Set
        LinkedStack<SpecialSkill> tempSet = new LinkedStack();

        Scanner scan = new Scanner(System.in);
        SpecialSkill selectionA;
        SpecialSkill selectionB;
        SpecialSkill skill = null;

        int choice;

        //select one of the skill
        selectionA = (SpecialSkill) skillSet.pop();
        selectionB = (SpecialSkill) skillSet.pop();

        System.out.println(name + ". Plese choose one of the skill for your game");
        System.out.println("1)Skill A    2)Skill B");

        do {
            System.out.print("Please enter 1/2 to chosse your skill : ");
            while (!scan.hasNextInt()) {
                System.out.print("Please Enter 1/2 to chosse your skill : ");
                scan.next();
            }
            choice = scan.nextInt();

            // simulation draw card the card not seleted put in the bottom of set
            // the previous cardset will put on top of the not selected card
            // always last in first out
            if (choice == 1) {
                skill = selectionA;
                tempSet.push(selectionB);
                tempSet.combine(skillSet);
                skillSet = tempSet;
                tempSet = null;

            } else if (choice == 2) {
                skill = selectionB;
                tempSet.push(selectionA);
                tempSet.combine(skillSet);
                skillSet = tempSet;
                tempSet = null;

            }
        } while (choice < 1 || choice > 2);

        //Get a skill
        System.out.printf("\n\nCongragulation! This is a Grade %c Skill\n", skill.getGrade())
        System.out.printf("This is the skill you get ** %s ** \n", skill.getName());
        System.out.printf("        %s \n", skill.getDescription());
        System.out.printf("Use times : %d \n", skill.getUseTimes());
        System.out.printf("Card Remaining : %d \n", skillSet.size());

        return skill;

    }
```

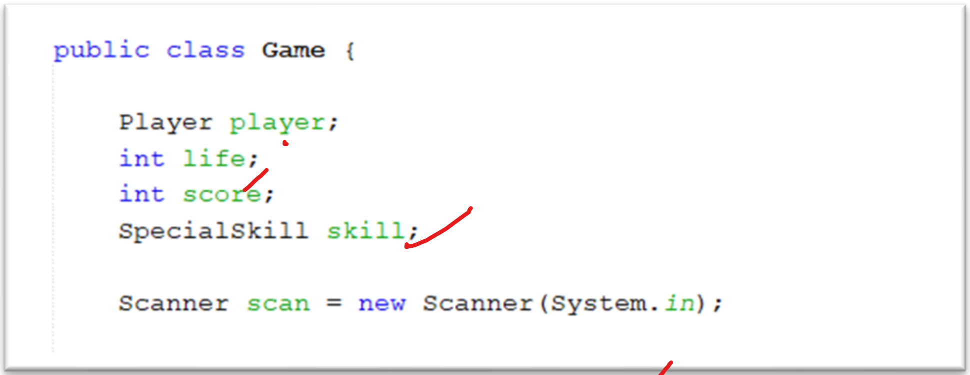
✓



```java
public static void resetSkillSet()
{
    skillSet.clear();
    initialSkillSet();
}
```

The **Game** class is the object will be doing control of game play for the Player. Each of the game will represent the player in the during the game process. Each player will get default 3 life, score start with 0 and a skill they choose.

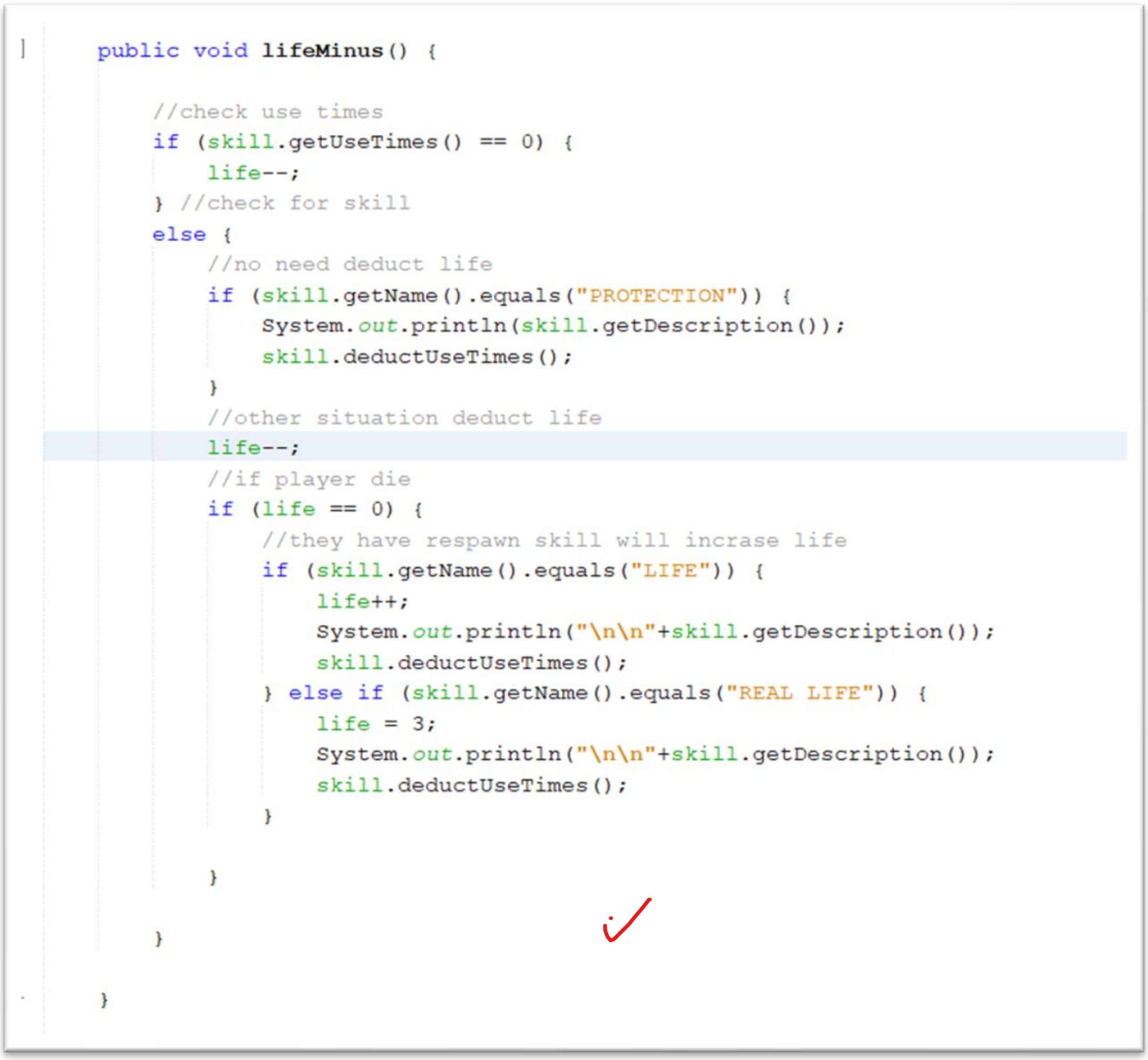
Game data field



```
public class Game {

    Player player;
    int life;
    int score;
    SpecialSkill skill;

    Scanner scan = new Scanner(System.in);
```

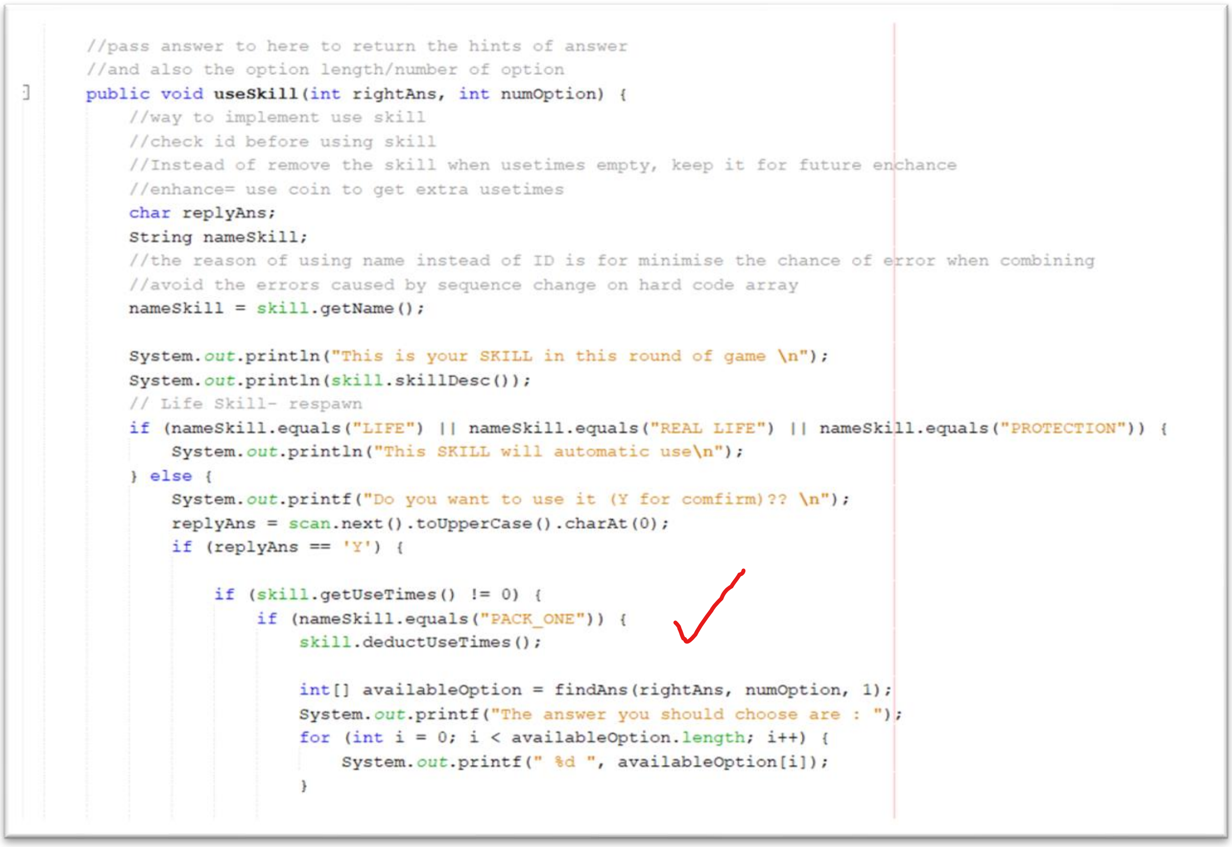
✓

Game function



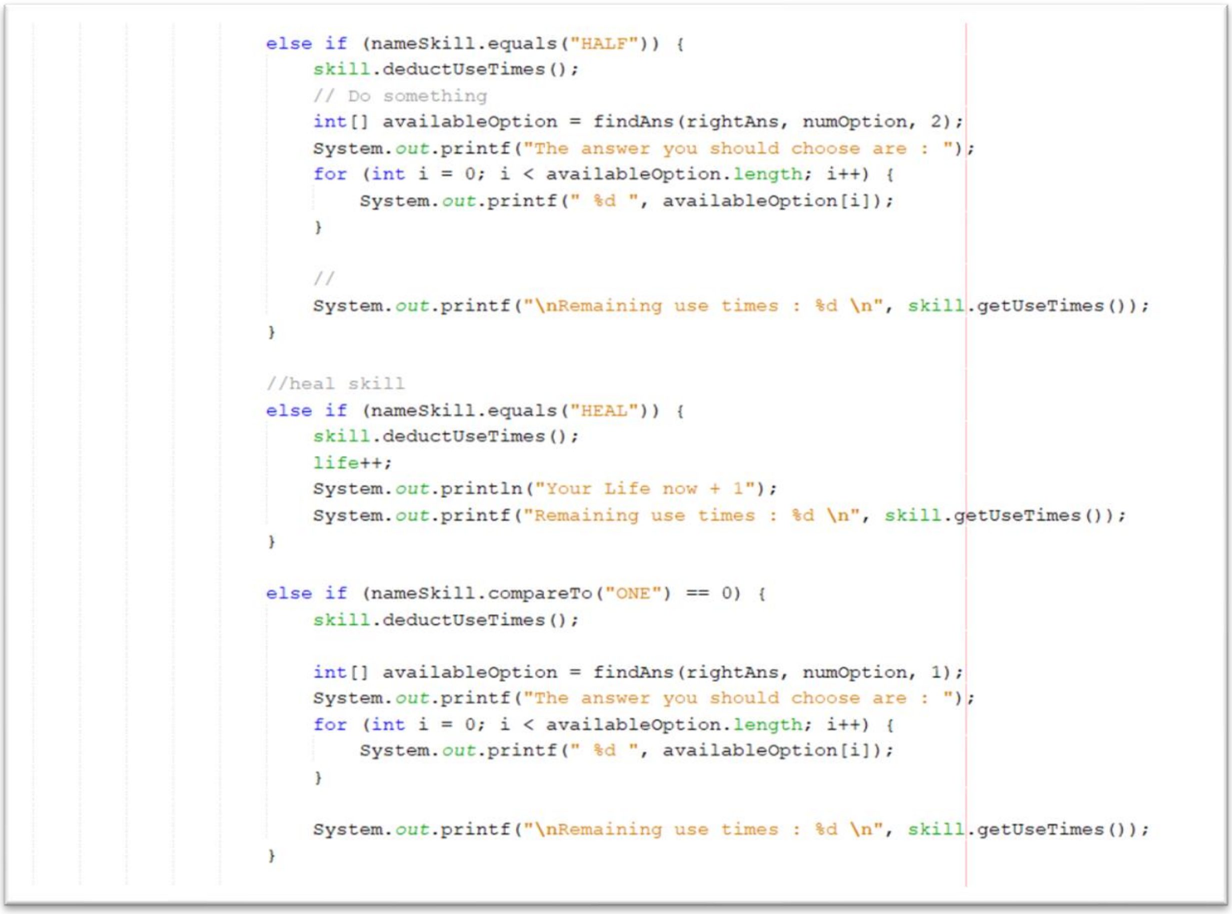
```
    public void lifeMinus() {

        //check use times
        if (skill.getUseTimes() == 0) {
            life--;
        } //check for skill
        else {
            //no need deduct life
            if (skill.getName().equals("PROTECTION")) {
                System.out.println(skill.getDescription());
                skill.deductUseTimes();
            }
            //other situation deduct life
            life--;
            //if player die
            if (life == 0) {
                //they have respawn skill will incrase life
                if (skill.getName().equals("LIFE")) {
                    life++;
                    System.out.println("\n\n"+skill.getDescription());
                    skill.deductUseTimes();
                } else if (skill.getName().equals("REAL LIFE")) {
                    life = 3;
                    System.out.println("\n\n"+skill.getDescription());
                    skill.deductUseTimes();
                }

            }

        }

    }
```

✓

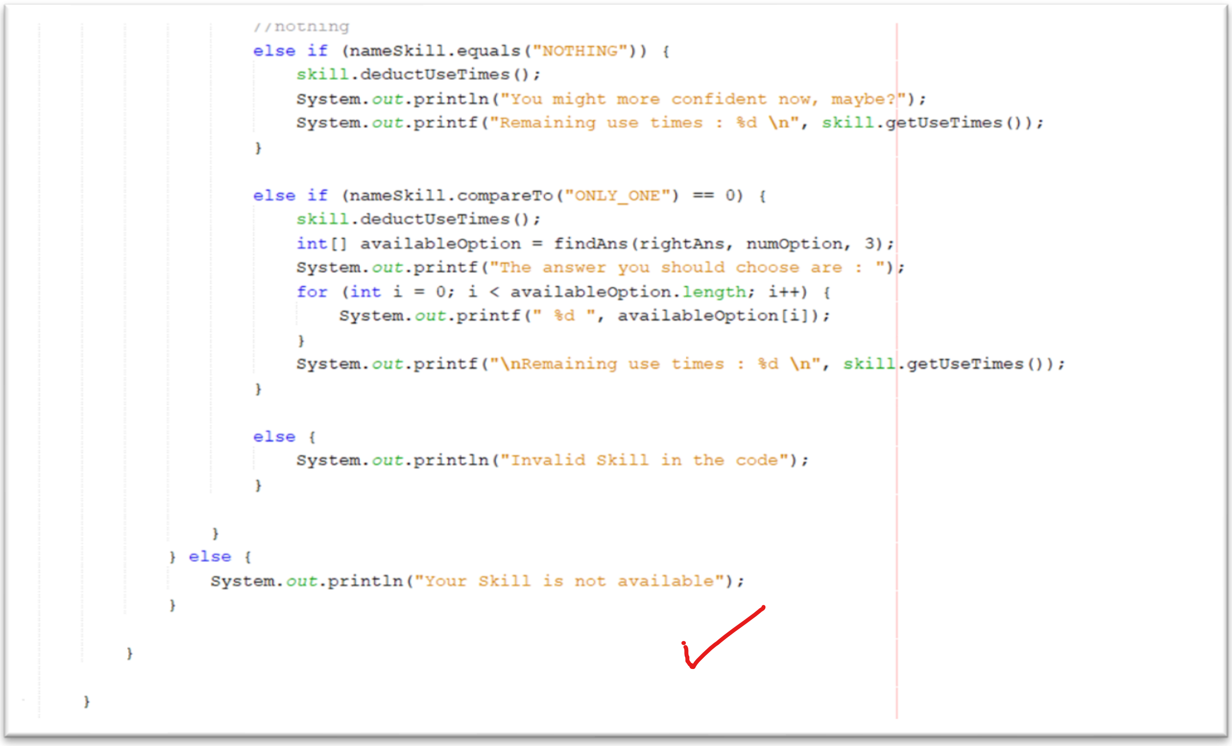


```
//pass answer to here to return the hints of answer
//and also the option length/number of option
public void useSkill(int rightAns, int numOption) {
    //way to implement use skill
    //check id before using skill
    //Instead of remove the skill when usetimes empty, keep it for future enchance
    //enhance= use coin to get extra usetimes
    char replyAns;
    String nameSkill;
    //the reason of using name instead of ID is for minimise the chance of error when combining
    //avoid the errors caused by sequence change on hard code array
    nameSkill = skill.getName();

    System.out.println("This is your SKILL in this round of game \n");
    System.out.println(skill.skillDesc());
    // Life Skill- respawn
    if (nameSkill.equals("LIFE") || nameSkill.equals("REAL LIFE") || nameSkill.equals("PROTECTION")) {
        System.out.println("This SKILL will automatic use\n");
    } else {
        System.out.printf("Do you want to use it (Y for comfirm)?? \n");
        replyAns = scan.next().toUpperCase().charAt(0);
        if (replyAns == 'Y') {

            if (skill.getUseTimes() != 0) {
                if (nameSkill.equals("PACK_ONE")) {
                    skill.deductUseTimes();

                    int[] availableOption = findAns(rightAns, numOption, 1);
                    System.out.printf("The answer you should choose are : ");
                    for (int i = 0; i < availableOption.length; i++) {
                        System.out.printf(" %d ", availableOption[i]);
                    }
```

✓

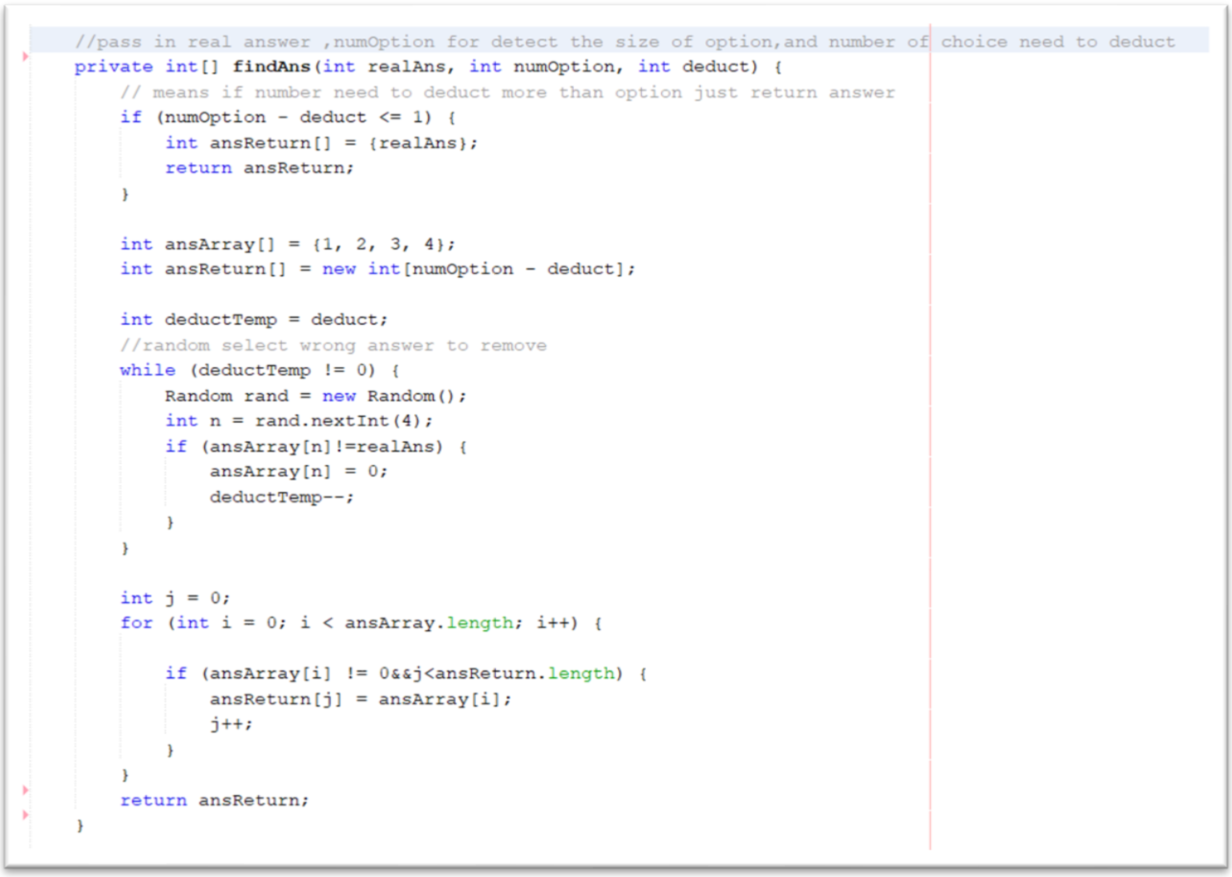


```
                else if (nameSkill.equals("HALF")) {
                    skill.deductUseTimes();
                    // Do something
                    int[] availableOption = findAns(rightAns, numOption, 2);
                    System.out.printf("The answer you should choose are : ");
                    for (int i = 0; i < availableOption.length; i++) {
                        System.out.printf(" %d ", availableOption[i]);
                    }

                    //
                    System.out.printf("\nRemaining use times : %d \n", skill.getUseTimes());
                }

                //heal skill
                else if (nameSkill.equals("HEAL")) {
                    skill.deductUseTimes();
                    life++;
                    System.out.println("Your Life now + 1");
                    System.out.printf("Remaining use times : %d \n", skill.getUseTimes());
                }

                else if (nameSkill.compareTo("ONE") == 0) {
                    skill.deductUseTimes();

                    int[] availableOption = findAns(rightAns, numOption, 1);
                    System.out.printf("The answer you should choose are : ");
                    for (int i = 0; i < availableOption.length; i++) {
                        System.out.printf(" %d ", availableOption[i]);
                    }

                    System.out.printf("\nRemaining use times : %d \n", skill.getUseTimes());
                }
```



```
                //nothing
                else if (nameSkill.equals("NOTHING")) {
                    skill.deductUseTimes();
                    System.out.println("You might more confident now, maybe?");
                    System.out.printf("Remaining use times : %d \n", skill.getUseTimes());
                }

                else if (nameSkill.compareTo("ONLY_ONE") == 0) {
                    skill.deductUseTimes();
                    int[] availableOption = findAns(rightAns, numOption, 3);
                    System.out.printf("The answer you should choose are : ");
                    for (int i = 0; i < availableOption.length; i++) {
                        System.out.printf(" %d ", availableOption[i]);
                    }
                    System.out.printf("\nRemaining use times : %d \n", skill.getUseTimes());
                }

                else {
                    System.out.println("Invalid Skill in the code");
                }

            }
        } else {
            System.out.println("Your Skill is not available");
        }

    }

}
```

✓



```
    //pass in real answer ,numOption for detect the size of option,and number of choice need to deduct
    private int[] findAns(int realAns, int numOption, int deduct) {
        // means if number need to deduct more than option just return answer
        if (numOption - deduct <= 1) {
            int ansReturn[] = {realAns};
            return ansReturn;
        }

        int ansArray[] = {1, 2, 3, 4};
        int ansReturn[] = new int[numOption - deduct];

        int deductTemp = deduct;
        //random select wrong answer to remove
        while (deductTemp != 0) {
            Random rand = new Random();
            int n = rand.nextInt(4);
            if (ansArray[n]!=realAns) {
                ansArray[n] = 0;
                deductTemp--;
            }
        }

        int j = 0;
        for (int i = 0; i < ansArray.length; i++) {

            if (ansArray[i] != 0&&j<ansReturn.length) {
                ansReturn[j] = ansArray[i];
                j++;
            }
        }
        return ansReturn;
    }
```

5. ADT Specification

ADT: **Stack**
Stack is a collection ADT of entries of a type T which allows duplicate elements inserted and removed according to the last in first out (LIFO) principle, this ADT will use the linked implementation to achieve best effort of using memory.

boolean push (T **newItem**)

Description	: Add **newItem** to the top of the stack.
Postcondition	: The **newItem** has been added to the top of the stack.
Return	: TRUE if successfully add the **newItem** to the top of the stack.

boolean pushAll (T [ ] **newItems**)

Description	: Add all the entries in **newItems** array start from zero to the last index, one by one to the top of the stack.
Postcondition	: All the entries in **newItems** array has been added to the top of stack accordingly.
Return	: TRUE if successfully added all the entries in array **newItems** to the top of the stack.

T pop ()

Description	: Remove the top entry of the stack.
Postcondition	: The entry at top of the stack has been removed from the stack.
Return	: The entry was removed from the stack, if the stack was empty return NULL.

T getTop ()

Description	: Retrieves the top entry of the stack.
Postcondition	: The stack remains unchanged.
Return	: The entry was retrieved from top of the stack.

int size ()

Description	: Retrieves the current size of the stack.
Postcondition	: The stack remains unchanged.
Return	: The size of the stack.

boolean isEmtpy ()

Description	: Determine whether the stack is empty.
Postcondition	: The stack remains unchanged.
Return	: TRUE if the stack is empty.

boolean clear ()

Description	: Remove all the entries in the stack.
Postcondition	: The stack has been clear and become empty.
Return	: TRUE if successfully remove all the entries in the stack.

boolean contain (T **anItem**)

Description	: Determine whether **anItem** exists in the stack.
Postcondition	: The stack remains unchanged.
Return	: TRUE if **anItem** exists in the stack.

int posOfElement (T **anItem**)

Description	: Retrieve the position of **anItem** in the stack.
Postcondition	: The stack remains unchanged.
Return	: The Integer array which contains positions of **anItem** locate, return empty array if **anItem** not exist in array.

T elementAtPos (int **position**)

Description	: Retrieve the entry in the stack with the specified **position**.
Postcondition	: The stack remains unchanged.
Return	: The entry was retrieved from a specific **position** of the stack.

boolean combine (LinkedStack **secondStack**)

Description	: Add the **secondStack** elements into the top of the current stack.
Postcondition	: The **secondStack** has been added into the stack
Return	: TRUE if successfully combine the stack with **secondStack**.

boolean equals (LinkedStack **secondStack**)

Description	: Check the stack and **secondStack** is the same or not.
Postcondition	: The stack remains unchanged.
Return	: TRUE if the stack with **secondStack** are the same.

boolean copy (LinkedStack **secondStack**)

Description	: The stack will clear and copy all the entries of the **secondStack** into itself.
Postcondition	: The stack has contained all the entries in the **secondStack**.
Return	: TRUE if stack successfully copy all the entries of the **secondStack**.

boolean copyPart (LinkedStack **secondStack**, int **num**)

Description	: The **stack** will clear and copy the entries of the **secondStack** from bottom to **num.**
Postcondition	: The stack has contained the entries in **secondStack** from bottom to **num**.
Return	: TRUE if stack successfully copy the entries of the **secondStack** from bottom to **num**.

boolean reverse ()

Description	: Reverse the order of entries in the stack.
Postcondition	: The order of entries in the stack has been reversed.
Return	: TRUE if successfully reverse the stack.

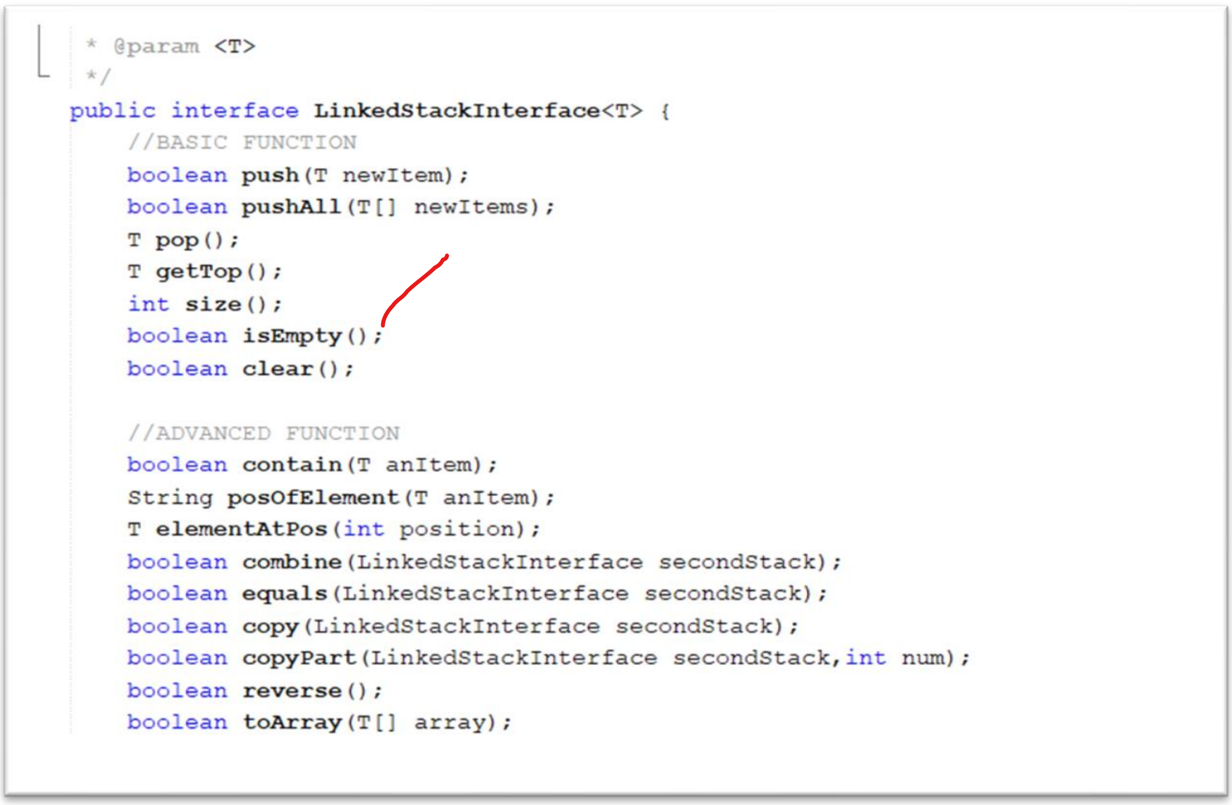
toArray (T[] **array**)

Description	: Copy the current stack element to the normal array style until the **array** full or the stack elements fully copy into **array**.
Postcondition	: The stack remains unchanged
Return	: TRUE if successfully copy into array

6. ADT Implementation

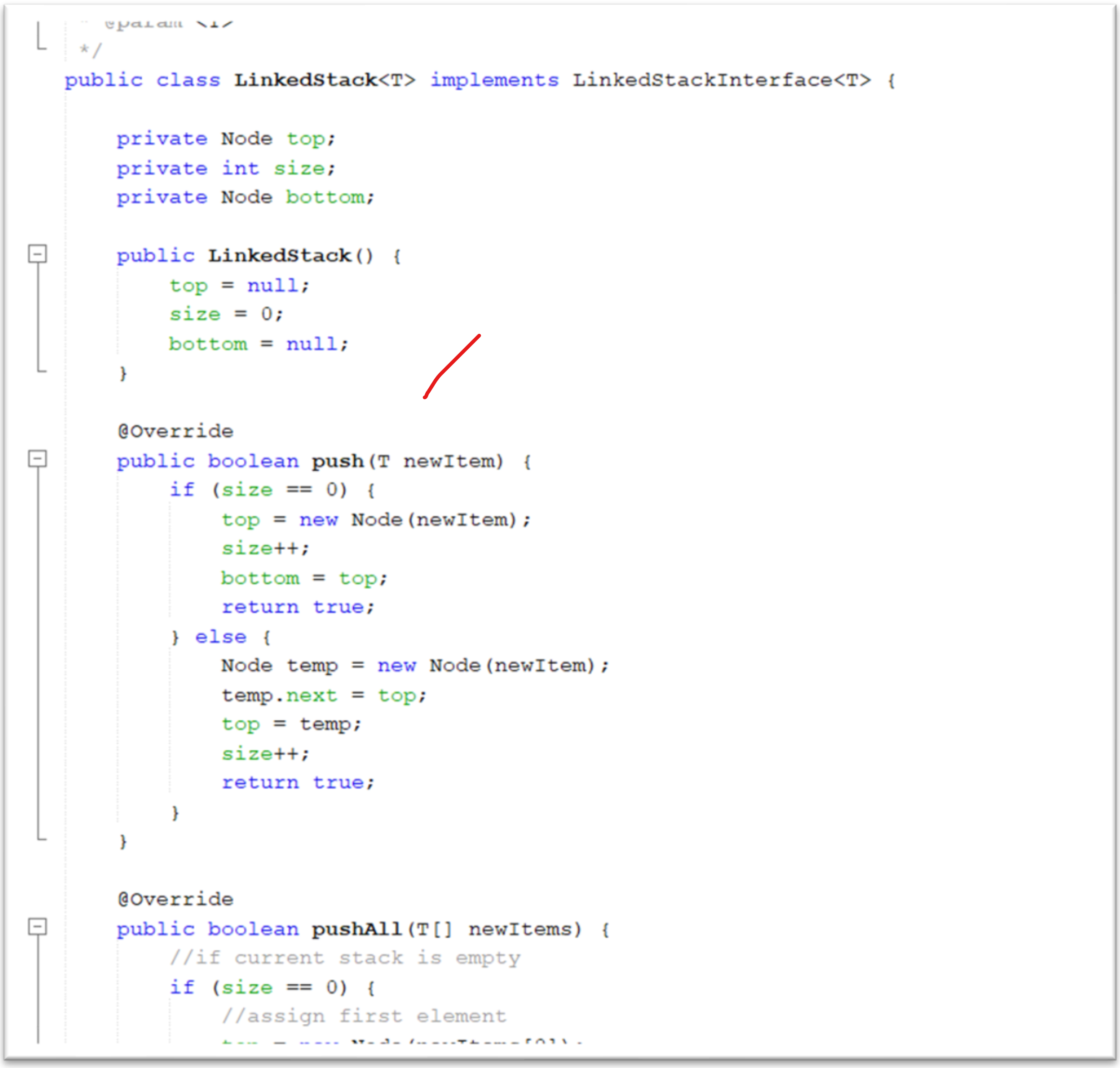
The collection of ADT stack is using the Linked implementation, and it will be called **LinkedStack**. It contains the basic function of common stack logic will do and also provide some advanced function for increase efficiency use of the stack.

LinkedStack Interface



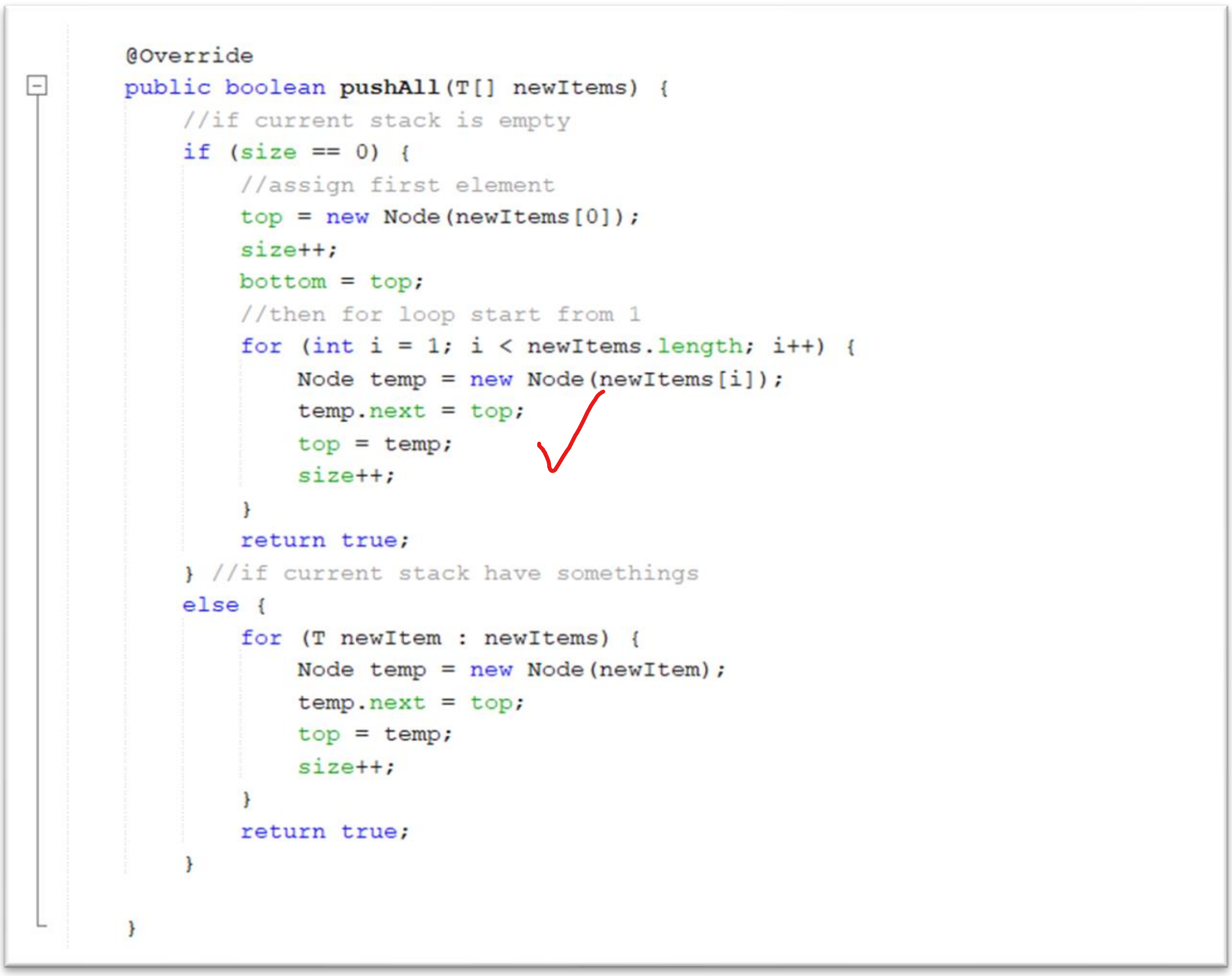
```
 * @param <T>
 */
public interface LinkedStackInterface<T> {
    //BASIC FUNCTION
    boolean push(T newItem);
    boolean pushAll(T[] newItems);
    T pop();
    T getTop();
    int size();
    boolean isEmpty();
    boolean clear();

    //ADVANCED FUNCTION
    boolean contain(T anItem);
    String posOfElement(T anItem);
    T elementAtPos(int position);
    boolean combine(LinkedStackInterface secondStack);
    boolean equals(LinkedStackInterface secondStack);
    boolean copy(LinkedStackInterface secondStack);
    boolean copyPart(LinkedStackInterface secondStack,int num);
    boolean reverse();
    boolean toArray(T[] array);
```

LinkedStack Classs



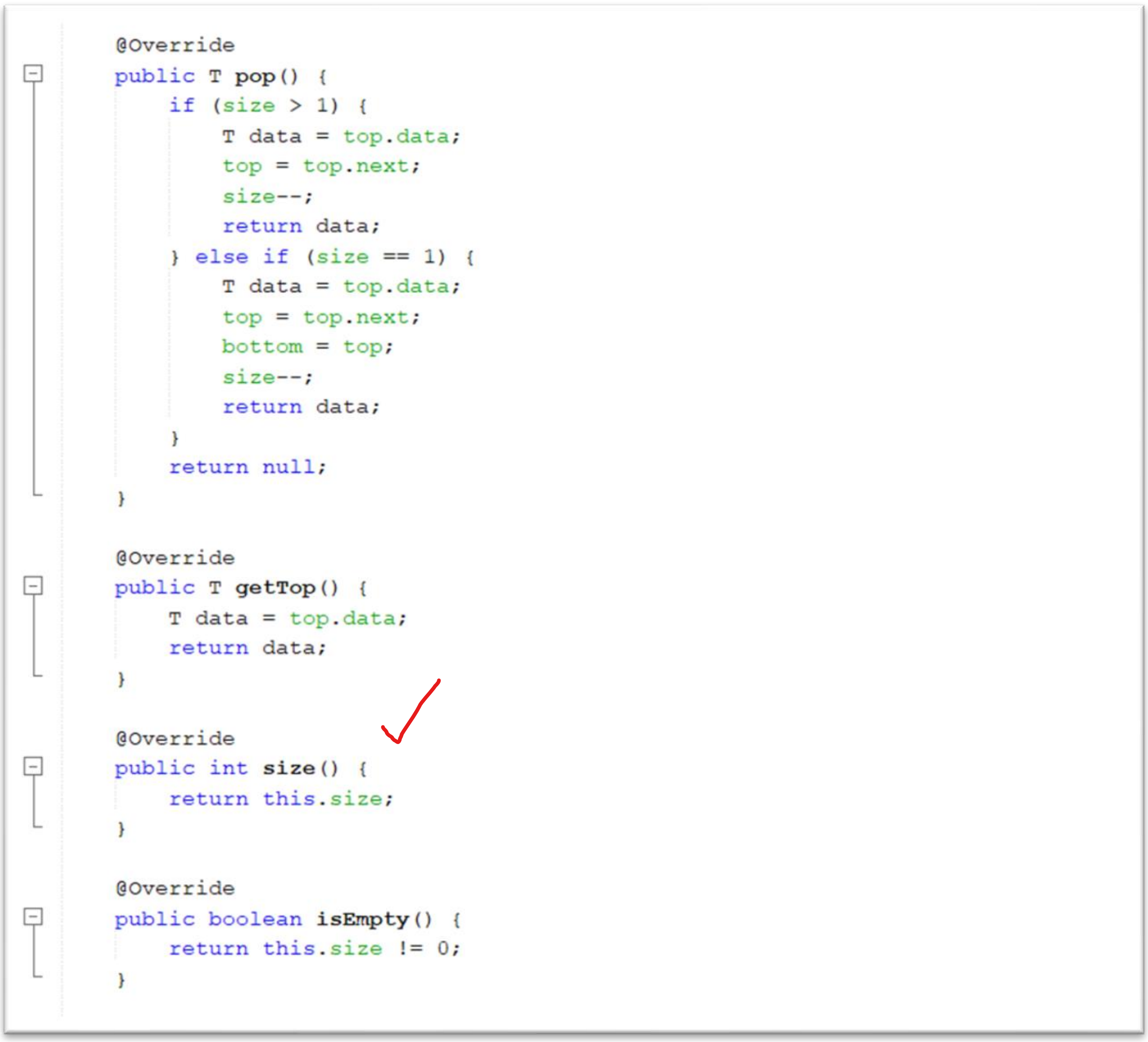
```
 * @param <T>
 */
public class LinkedStack<T> implements LinkedStackInterface<T> {

    private Node top;
    private int size;
    private Node bottom;

    public LinkedStack() {
        top = null;
        size = 0;
        bottom = null;
    }

    @Override
    public boolean push(T newItem) {
        if (size == 0) {
            top = new Node(newItem);
            size++;
            bottom = top;
            return true;
        } else {
            Node temp = new Node(newItem);
            temp.next = top;
            top = temp;
            size++;
            return true;
        }
    }

    @Override
    public boolean pushAll(T[] newItems) {
        //if current stack is empty
        if (size == 0) {
            //assign first element
            top = new Node(newItems[0]);
```

✓



```
@Override
public boolean pushAll(T[] newItems) {
    //if current stack is empty
    if (size == 0) {
        //assign first element
        top = new Node(newItems[0]);
        size++;
        bottom = top;
        //then for loop start from 1
        for (int i = 1; i < newItems.length; i++) {
            Node temp = new Node(newItems[i]);
            temp.next = top;
            top = temp;
            size++;
        }
        return true;
    } //if current stack have somethings
    else {
        for (T newItem : newItems) {
            Node temp = new Node(newItem);
            temp.next = top;
            top = temp;
            size++;
        }
        return true;
    }

}
```

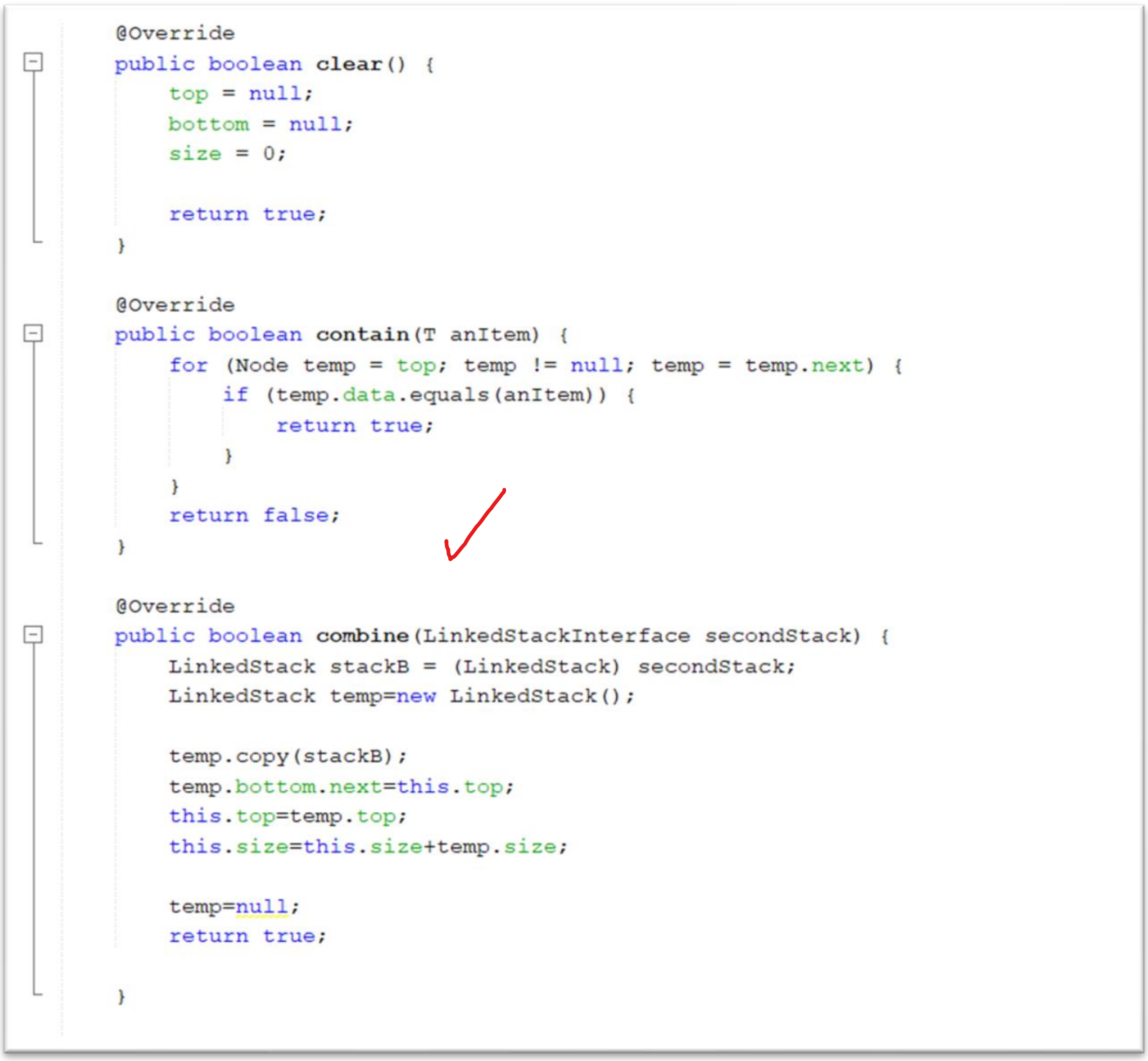
✓



```java
    @Override
    public T pop() {
        if (size > 1) {
            T data = top.data;
            top = top.next;
            size--;
            return data;
        } else if (size == 1) {
            T data = top.data;
            top = top.next;
            bottom = top;
            size--;
            return data;
        }
        return null;
    }

    @Override
    public T getTop() {
        T data = top.data;
        return data;
    }
```

✓

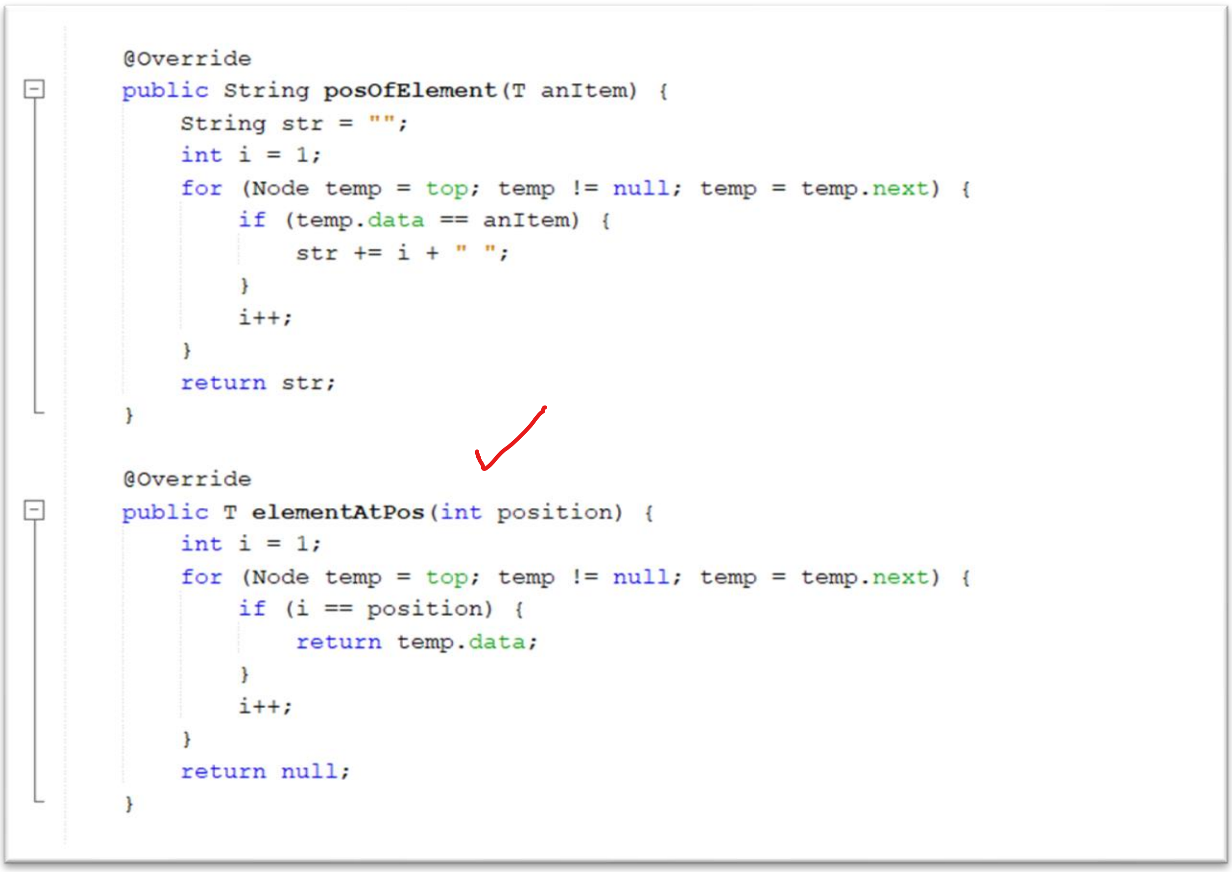
```java
    @Override
    public int size() {
        return this.size;
    }

    @Override
    public boolean isEmpty() {
        return this.size != 0;
    }
```



```
@Override
public boolean clear() {
    top = null;
    bottom = null;
    size = 0;

    return true;
}

@Override
public boolean contain(T anItem) {
    for (Node temp = top; temp != null; temp = temp.next) {
        if (temp.data.equals(anItem)) {
            return true;
        }
    }
    return false;
}
```

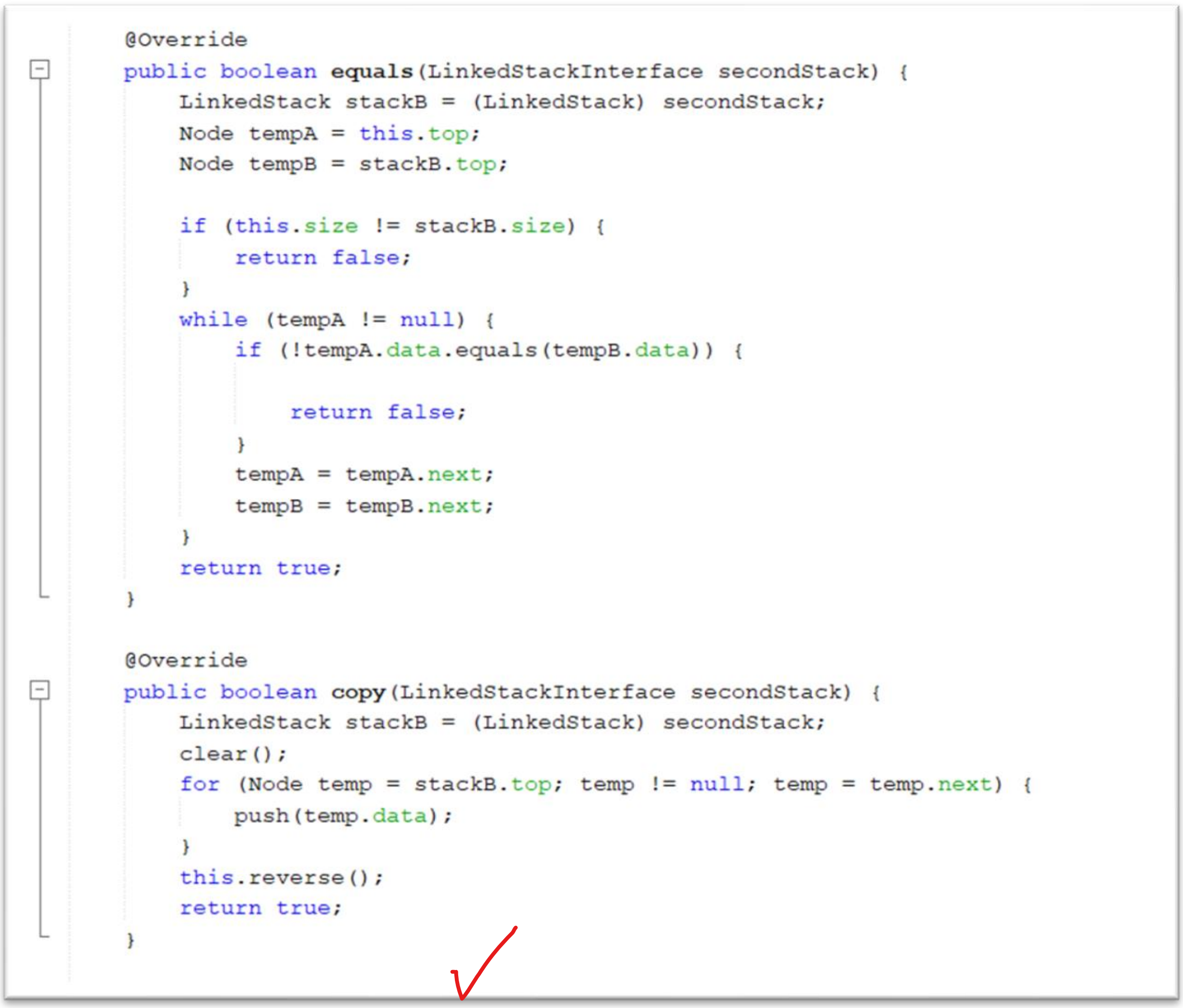
✓

```
@Override
public boolean combine(LinkedStackInterface secondStack) {
    LinkedStack stackB = (LinkedStack) secondStack;
    LinkedStack temp=new LinkedStack();

    temp.copy(stackB);
    temp.bottom.next=this.top;
    this.top=temp.top;
    this.size=this.size+temp.size;

    temp=null;
    return true;

}
```



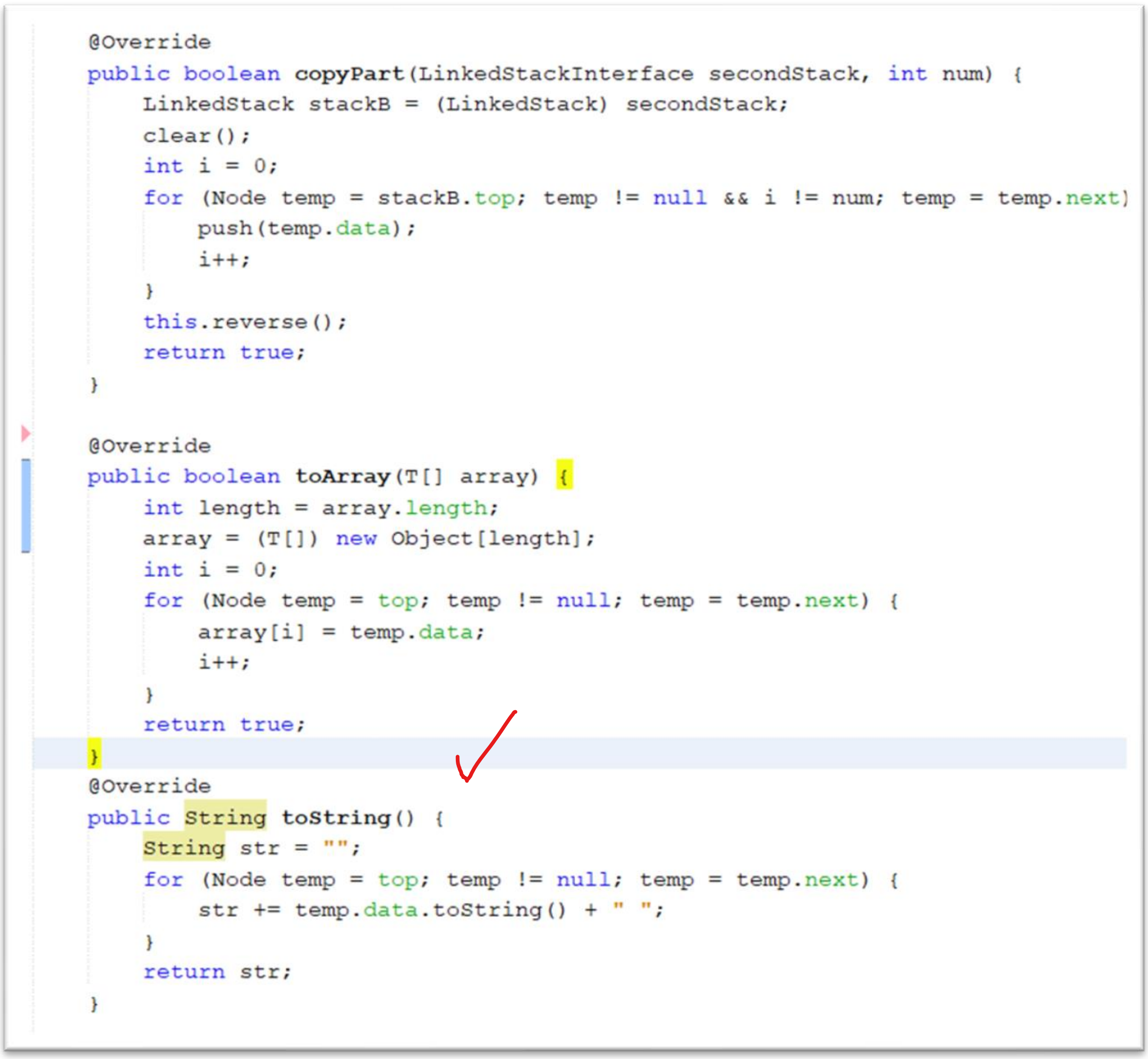
```
@Override
public String posOfElement(T anItem) {
    String str = "";
    int i = 1;
    for (Node temp = top; temp != null; temp = temp.next) {
        if (temp.data == anItem) {
            str += i + " ";
        }
        i++;
    }
    return str;
}

@Override
public T elementAtPos(int position) {
    int i = 1;
    for (Node temp = top; temp != null; temp = temp.next) {
        if (i == position) {
            return temp.data;
        }
        i++;
    }
    return null;
}
```

✓



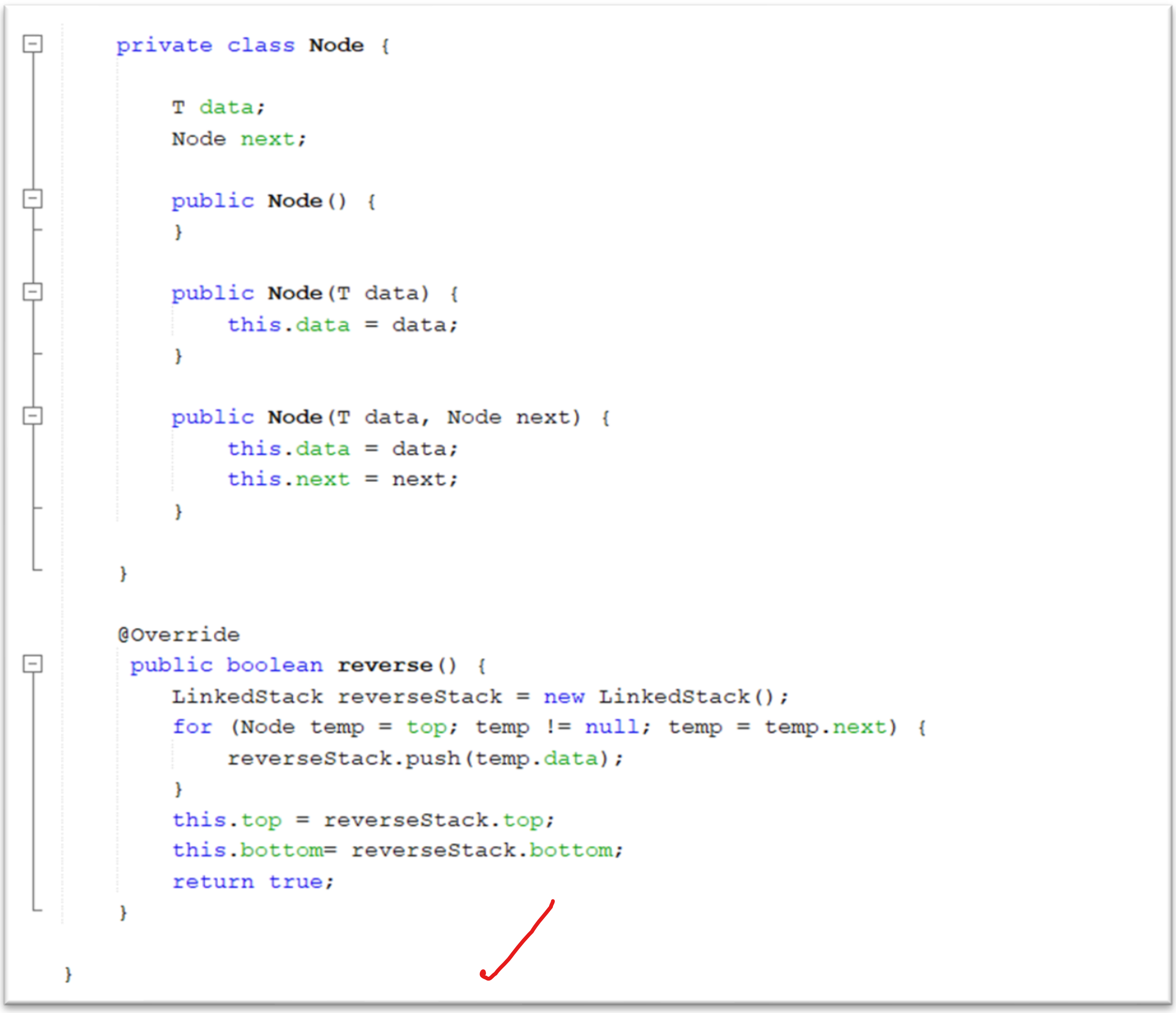
```
@Override
public boolean equals(LinkedStackInterface secondStack) {
    LinkedStack stackB = (LinkedStack) secondStack;
    Node tempA = this.top;
    Node tempB = stackB.top;

    if (this.size != stackB.size) {
        return false;
    }
    while (tempA != null) {
        if (!tempA.data.equals(tempB.data)) {

            return false;
        }
        tempA = tempA.next;
        tempB = tempB.next;
    }
    return true;
}

@Override
public boolean copy(LinkedStackInterface secondStack) {
    LinkedStack stackB = (LinkedStack) secondStack;
    clear();
    for (Node temp = stackB.top; temp != null; temp = temp.next) {
        push(temp.data);
    }
    this.reverse();
    return true;
}
```

✓



```
@Override
public boolean copyPart(LinkedStackInterface secondStack, int num) {
    LinkedStack stackB = (LinkedStack) secondStack;
    clear();
    int i = 0;
    for (Node temp = stackB.top; temp != null && i != num; temp = temp.next)
        push(temp.data);
        i++;
    }
    this.reverse();
    return true;
}

@Override
public boolean toArray(T[] array) {
    int length = array.length;
    array = (T[]) new Object[length];
    int i = 0;
    for (Node temp = top; temp != null; temp = temp.next) {
        array[i] = temp.data;
        i++;
    }
    return true;
}
@Override
public String toString() {
    String str = "";
    for (Node temp = top; temp != null; temp = temp.next) {
        str += temp.data.toString() + " ";
    }
    return str;
}
```

✓



```
    private class Node {

        T data;
        Node next;

        public Node() {
        }

        public Node(T data) {
            this.data = data;
        }

        public Node(T data, Node next) {
            this.data = data;
            this.next = next;
        }

    }

    @Override
     public boolean reverse() {
        LinkedStack reverseStack = new LinkedStack();
        for (Node temp = top; temp != null; temp = temp.next) {
            reverseStack.push(temp.data);
        }
        this.top = reverseStack.top;
        this.bottom= reverseStack.bottom;
        return true;
    }

}
```

✓

7. Selection of Collection ADTs

For the BigBrain quiz game, I am mainly handling the **SpecialSkill function and entity**, the ADT collection I choose for the SpecialSkill is **LinkedStack**, that a kind of stack with last in first out principle. The reason for using LinkedStack to store SpecialSkill is for implement the card set concept for the skill setting used in the gameplay of BigBrain. Every time we start a new game it will disrupt the sequence of hard code array data and push all of them into a card set just similar to the concept of playing poker.

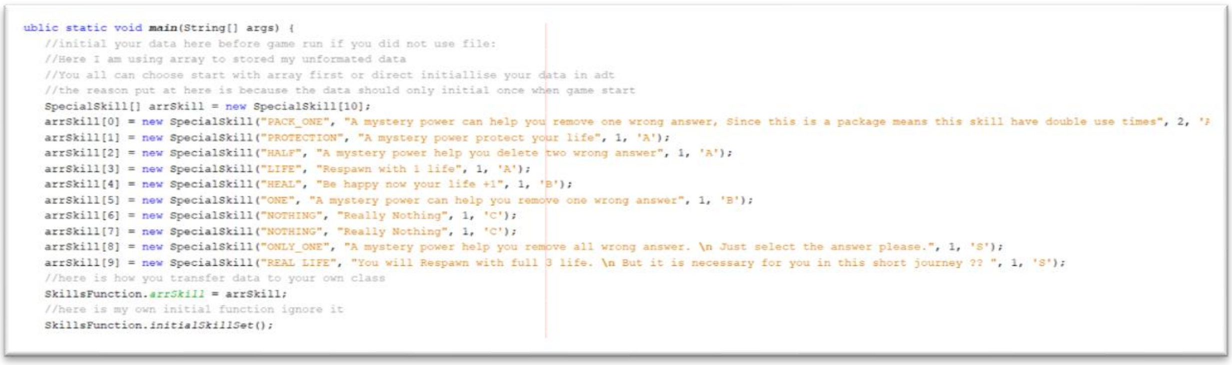
The push function can represent the action of inserting a cart into the card set, then the pop function can represent the distribution of the card from the card set. Since using the linked implementation can also easily achieve the combine function which is the concept of stack A can directly push to the top of the stack B. That means I can achieve the concept of putting the unselected card into the bottom of the card set.

Furthermore, another reason for me using linked implementation instead of using array implementation is because I hope that my stack can be flexible enough and minimize the resources and processing power as much as possible. Since the stack can consider a highly efficient way to store and process data, due to users no need to take care about the middle part of elements in the stack

Then, for the **Game** entity me and my team member decided to use the queue ADT collection. Which is allow the BigBrain handle the sequence player during game journey. We will enqueue all the participating player to the line, and dequeue them one by one for answer the quiz after player answered the quiz will enqueue into line again, if they run out of their life that will not able to join into the line already.

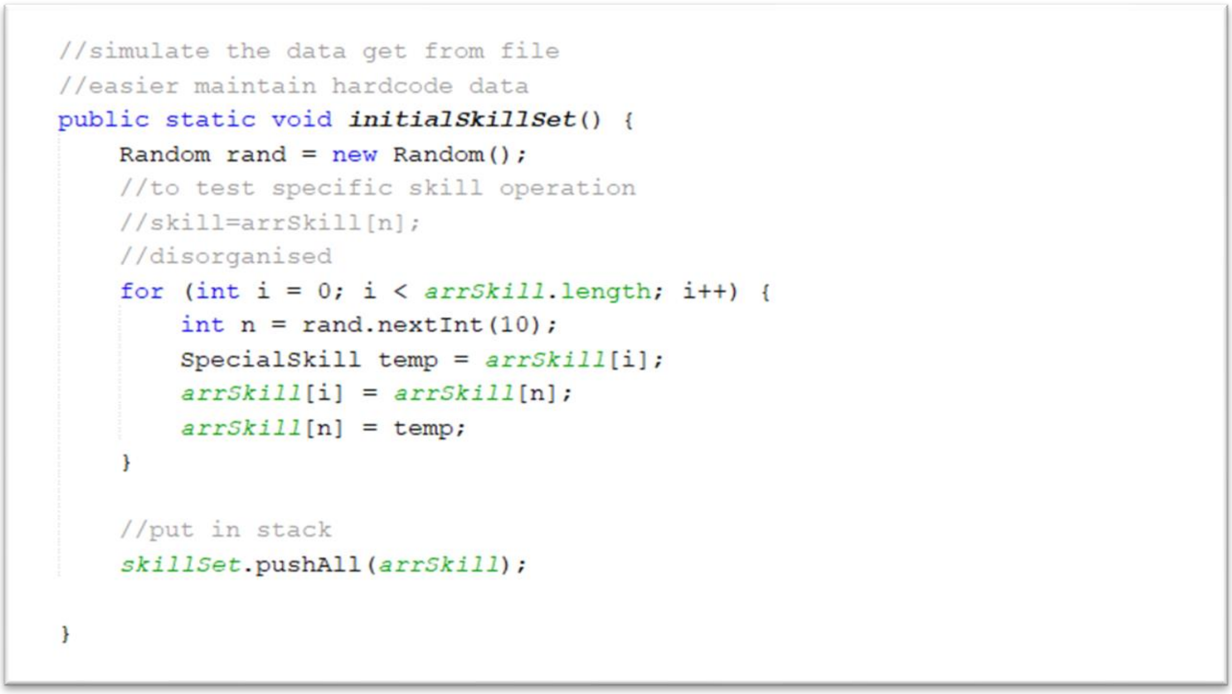
8. Implementation of Client Program

This is codes in the Driver.java for initial the SpecialSkill array



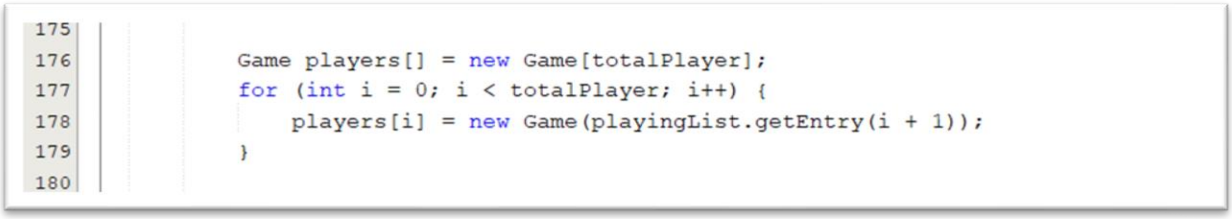
```java
ublic static void main(String[] args) {
    //initial your data here before game run if you did not use file:
    //Here I am using array to stored my unformated data
    //You all can choose start with array first or direct initiallise your data in adt
    //the reason put at here is because the data should only initial once when game start
    SpecialSkill[] arrSkill = new SpecialSkill[10];
    arrSkill[0] = new SpecialSkill("PACK_ONE", "A mystery power can help you remove one wrong answer, Since this is a package means this skill have double use times", 2, '
    arrSkill[1] = new SpecialSkill("PROTECTION", "A mystery power protect your life", 1, 'A');
    arrSkill[2] = new SpecialSkill("HALF", "A mystery power help you delete two wrong answer", 1, 'A');
    arrSkill[3] = new SpecialSkill("LIFE", "Respawn with 1 life", 1, 'A');
    arrSkill[4] = new SpecialSkill("HEAL", "Be happy now your life +1", 1, 'B');
    arrSkill[5] = new SpecialSkill("ONE", "A mystery power can help you remove one wrong answer", 1, 'B');
    arrSkill[6] = new SpecialSkill("NOTHING", "Really Nothing", 1, 'C');
    arrSkill[7] = new SpecialSkill("NOTHING", "Really Nothing", 1, 'C');
    arrSkill[8] = new SpecialSkill("ONLY_ONE", "A mystery power help you remove all wrong answer. \n Just select the answer please.", 1, 'S');
    arrSkill[9] = new SpecialSkill("REAL LIFE", "You will Respawn with full 3 life. \n But it is necessary for you in this short journey ?? ", 1, 'S');
    //here is how you transfer data to your own class
    SkillsFunction.arrSkill = arrSkill;
    //here is my own initial function ignore it
    SkillsFunction.initialSkillSet();
```

Here is inside the SkillsFunction. After initial the SpecialSkill array it will call the initialSkillSet function for disrupt the sequence of SpecialSkill array and push all the elements of the array into skillSet (LinkedStack).



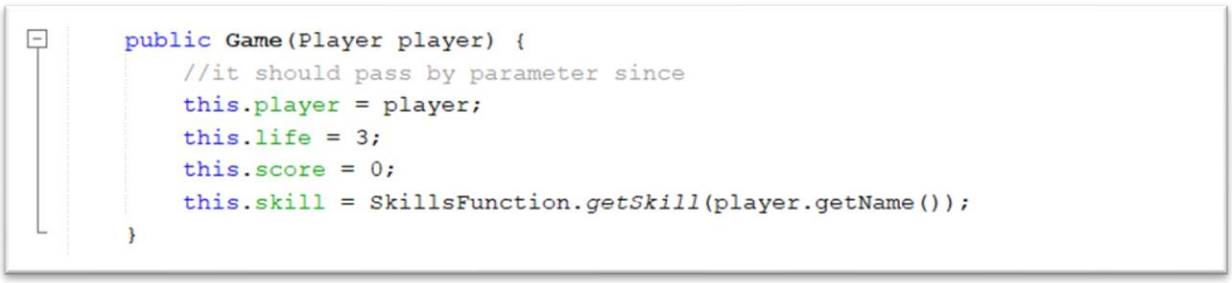
```java
//simulate the data get from file
//easier maintain hardcode data
public static void initialSkillSet() {
    Random rand = new Random();
    //to test specific skill operation
    //skill=arrSkill[n];
    //disorganised
    for (int i = 0; i < arrSkill.length; i++) {
        int n = rand.nextInt(10);
        SpecialSkill temp = arrSkill[i];
        arrSkill[i] = arrSkill[n];
        arrSkill[n] = temp;
    }

    //put in stack
    skillSet.pushAll(arrSkill);

}
```

Here is inside the Driver.java. After initial done when the program creates the new Game entity for set the game control for each of the player. It will let the players to choose a skill.



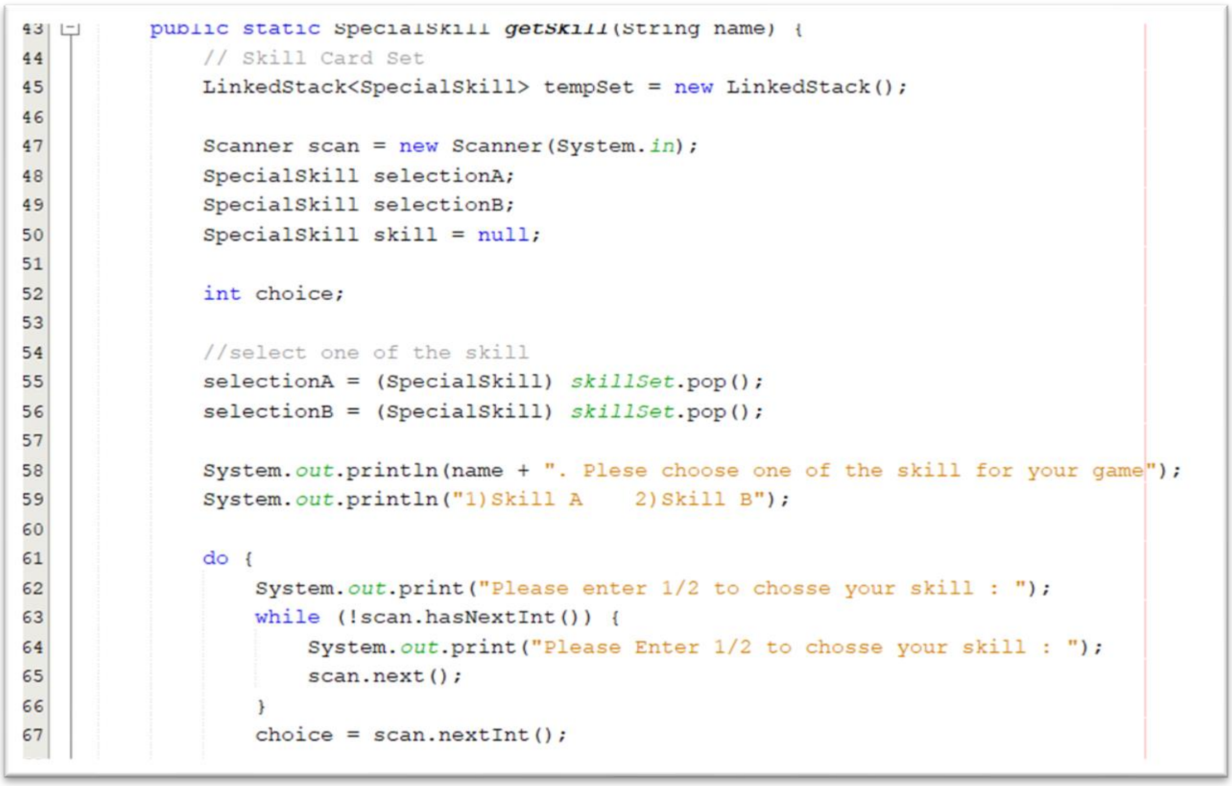
```java

        Game players[] = new Game[totalPlayer];
        for (int i = 0; i < totalPlayer; i++) {
            players[i] = new Game(playingList.getEntry(i + 1));
        }

```

Here is inside the game.java. Every new object of Game created it will call the function to get a SpecialSkill for this object.



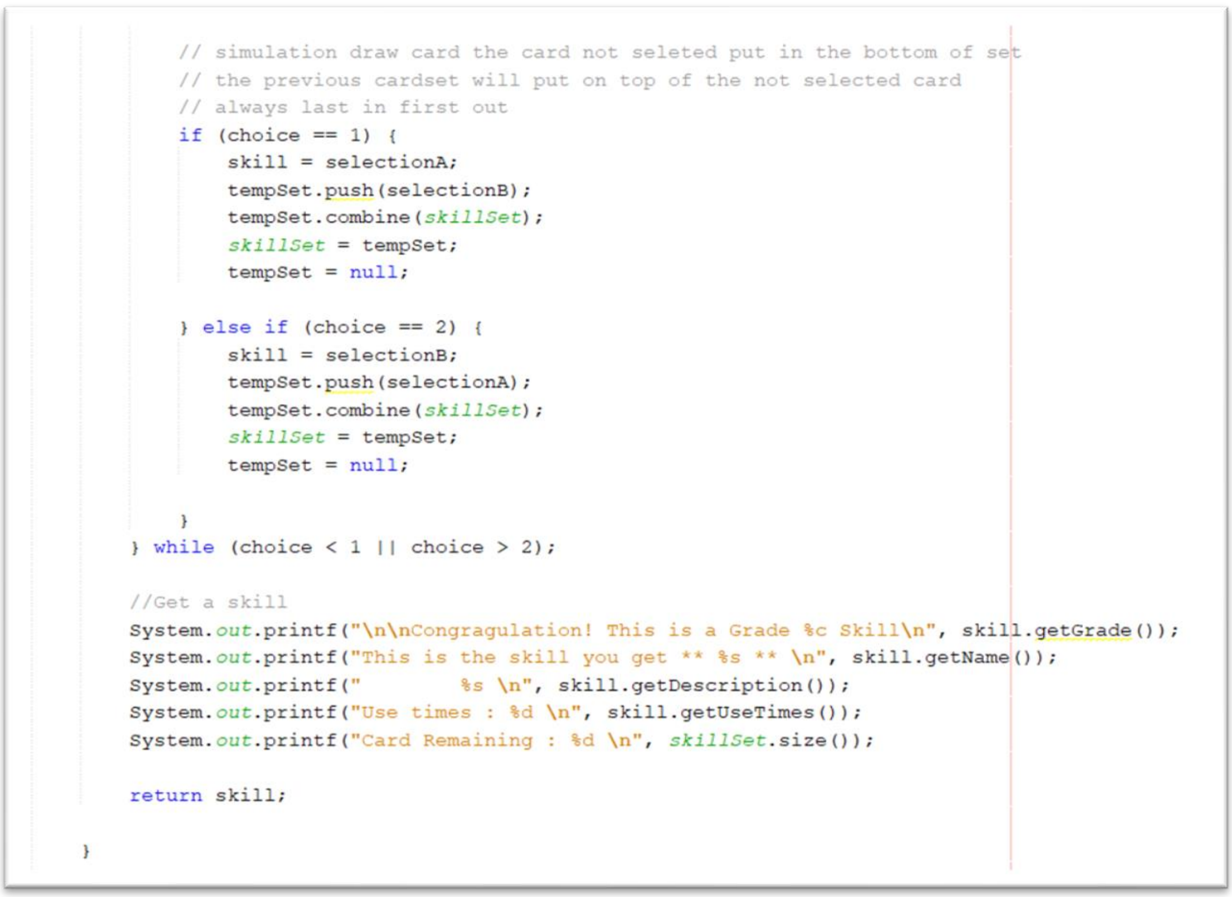
```java
public Game(Player player) {
    //it should pass by parameter since
    this.player = player;
    this.life = 3;
    this.score = 0;
    this.skill = SkillsFunction.getSkill(player.getName());
}
```

Here is inside the SkillsFunction. This is getSkill function will call when the new object of Game created. This is the function will pop the skill in the skillSet for player to choose a skill for gameplay



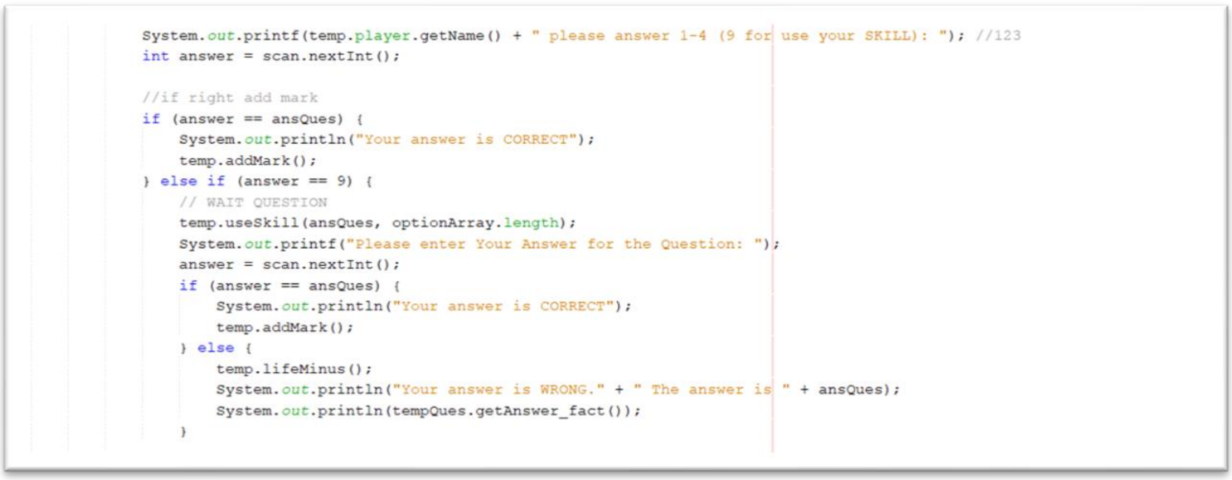
```java
    public static SpecialSkill getSkill(String name) {
        // Skill Card Set
        LinkedStack<SpecialSkill> tempSet = new LinkedStack();

        Scanner scan = new Scanner(System.in);
        SpecialSkill selectionA;
        SpecialSkill selectionB;
        SpecialSkill skill = null;

        int choice;

        //select one of the skill
        selectionA = (SpecialSkill) skillSet.pop();
        selectionB = (SpecialSkill) skillSet.pop();

        System.out.println(name + ". Plese choose one of the skill for your game");
        System.out.println("1)Skill A    2)Skill B");

        do {
            System.out.print("Please enter 1/2 to chosse your skill : ");
            while (!scan.hasNextInt()) {
                System.out.print("Please Enter 1/2 to chosse your skill : ");
                scan.next();
            }
            choice = scan.nextInt();
```

.



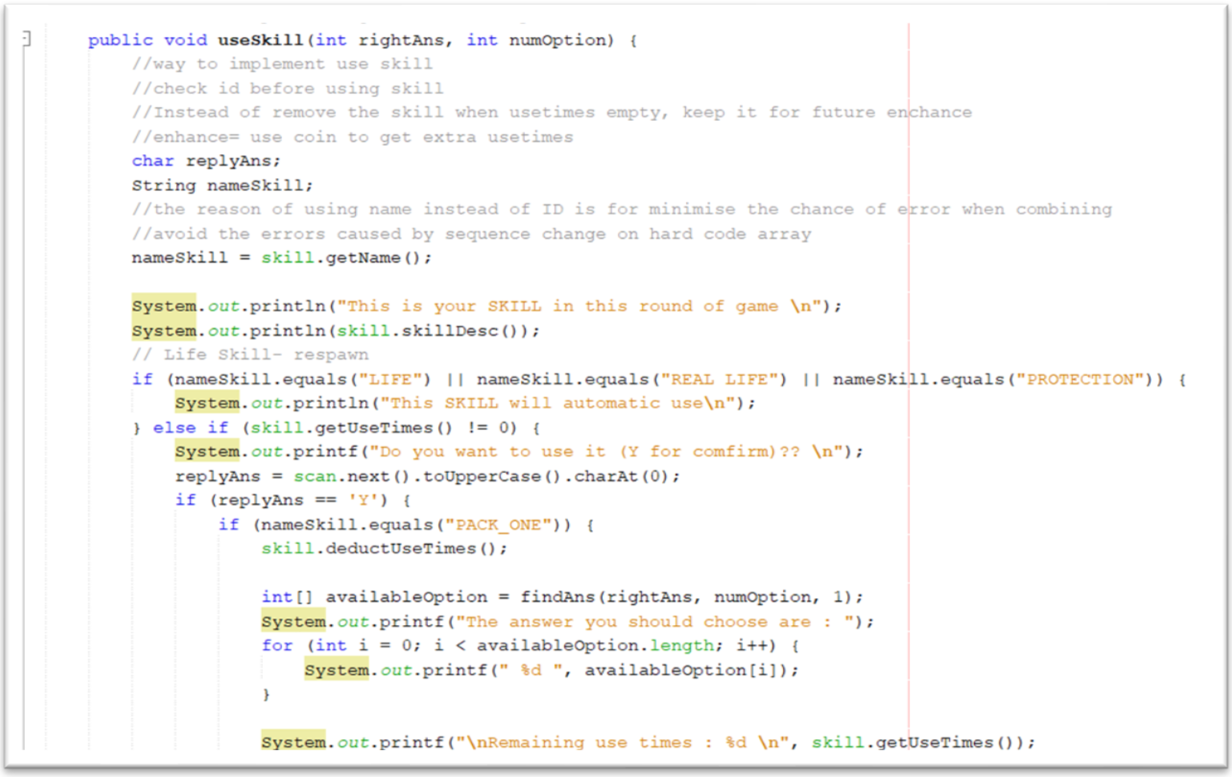
```java
            // simulation draw card the card not seleted put in the bottom of set
            // the previous cardset will put on top of the not selected card
            // always last in first out
            if (choice == 1) {
                skill = selectionA;
                tempSet.push(selectionB);
                tempSet.combine(skillSet);
                skillSet = tempSet;
                tempSet = null;

            } else if (choice == 2) {
                skill = selectionB;
                tempSet.push(selectionA);
                tempSet.combine(skillSet);
                skillSet = tempSet;
                tempSet = null;

            }
        } while (choice < 1 || choice > 2);

        //Get a skill
        System.out.printf("\n\nCongragulation! This is a Grade %c Skill\n", skill.getGrade());
        System.out.printf("This is the skill you get ** %s ** \n", skill.getName());
        System.out.printf("        %s \n", skill.getDescription());
        System.out.printf("Use times : %d \n", skill.getUseTimes());
        System.out.printf("Card Remaining : %d \n", skillSet.size());

        return skill;

    }
```

Here is inside the driver when player answering their question they can enter ‘9’ for using their skill for help when their skill is available.

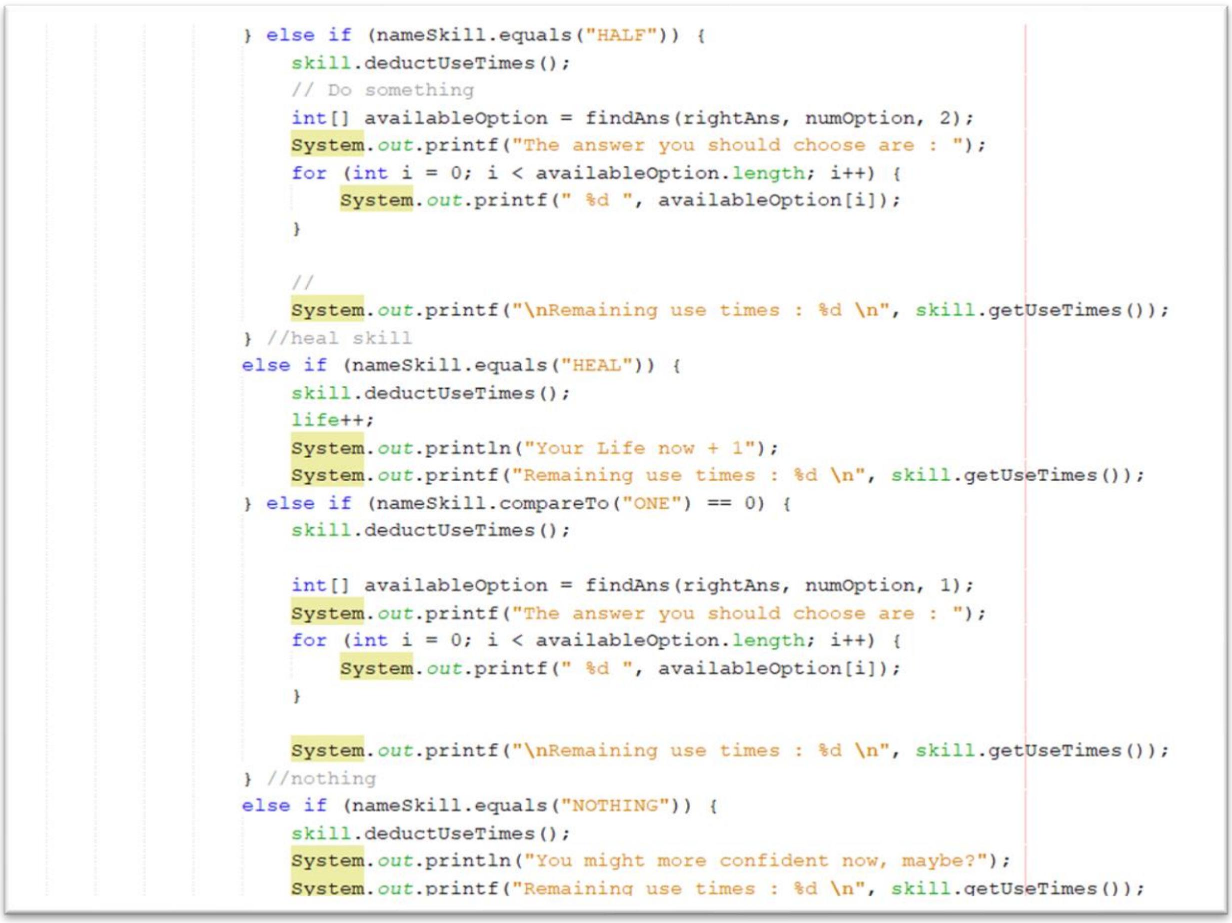


```java
System.out.printf(temp.player.getName() + " please answer 1-4 (9 for use your SKILL): "); //123
int answer = scan.nextInt();

//if right add mark
if (answer == ansQues) {
    System.out.println("Your answer is CORRECT");
    temp.addMark();
} else if (answer == 9) {
    // WAIT QUESTION
    temp.useSkill(ansQues, optionArray.length);
    System.out.printf("Please enter Your Answer for the Question: ");
    answer = scan.nextInt();
    if (answer == ansQues) {
        System.out.println("Your answer is CORRECT");
        temp.addMark();
    } else {
        temp.lifeMinus();
        System.out.println("Your answer is WRONG." + " The answer is " + ansQues);
        System.out.println(tempQues.getAnswer_fact());
    }
```

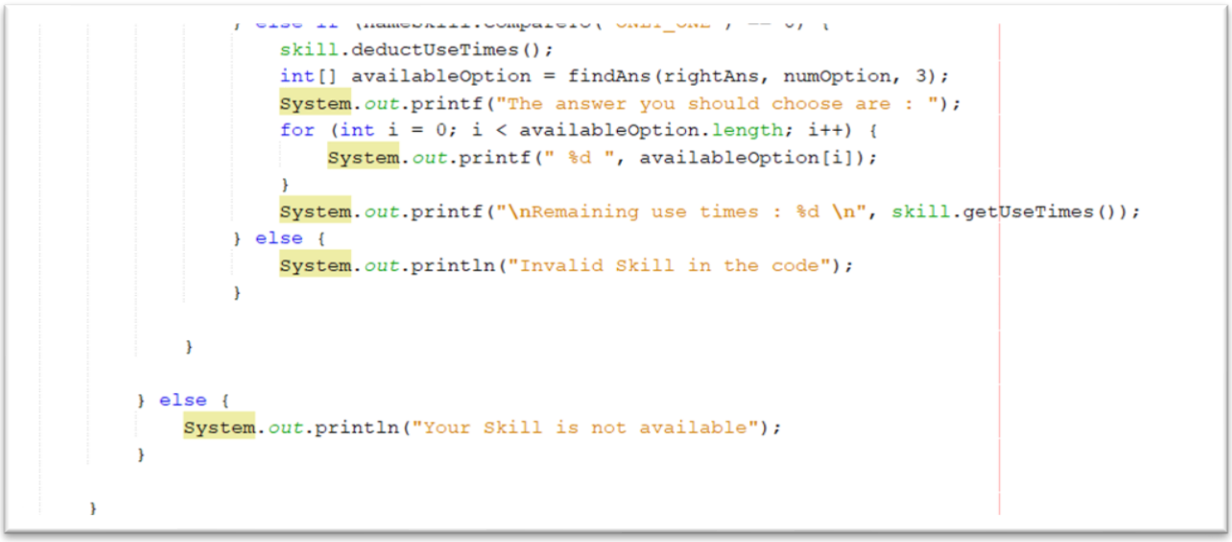
Here is inside the Gam.java. This is the useSkill function will show the skill information and also proceed the logic of using skill, with pass in the right answer and the option of the answer for the specific question to do the necessary calculation for provide help for player.



```java
public void useSkill(int rightAns, int numOption) {
    //way to implement use skill
    //check id before using skill
    //Instead of remove the skill when usetimes empty, keep it for future enchance
    //enhance= use coin to get extra usetimes
    char replyAns;
    String nameSkill;
    //the reason of using name instead of ID is for minimise the chance of error when combining
    //avoid the errors caused by sequence change on hard code array
    nameSkill = skill.getName();

    System.out.println("This is your SKILL in this round of game \n");
    System.out.println(skill.skillDesc());
    // Life Skill- respawn
    if (nameSkill.equals("LIFE") || nameSkill.equals("REAL LIFE") || nameSkill.equals("PROTECTION")) {
        System.out.println("This SKILL will automatic use\n");
    } else if (skill.getUseTimes() != 0) {
        System.out.printf("Do you want to use it (Y for comfirm)?? \n");
        replyAns = scan.next().toUpperCase().charAt(0);
        if (replyAns == 'Y') {
            if (nameSkill.equals("PACK_ONE")) {
                skill.deductUseTimes();

                int[] availableOption = findAns(rightAns, numOption, 1);
                System.out.printf("The answer you should choose are : ");
                for (int i = 0; i < availableOption.length; i++) {
                    System.out.printf(" %d ", availableOption[i]);
                }

                System.out.printf("\nRemaining use times : %d \n", skill.getUseTimes());
```

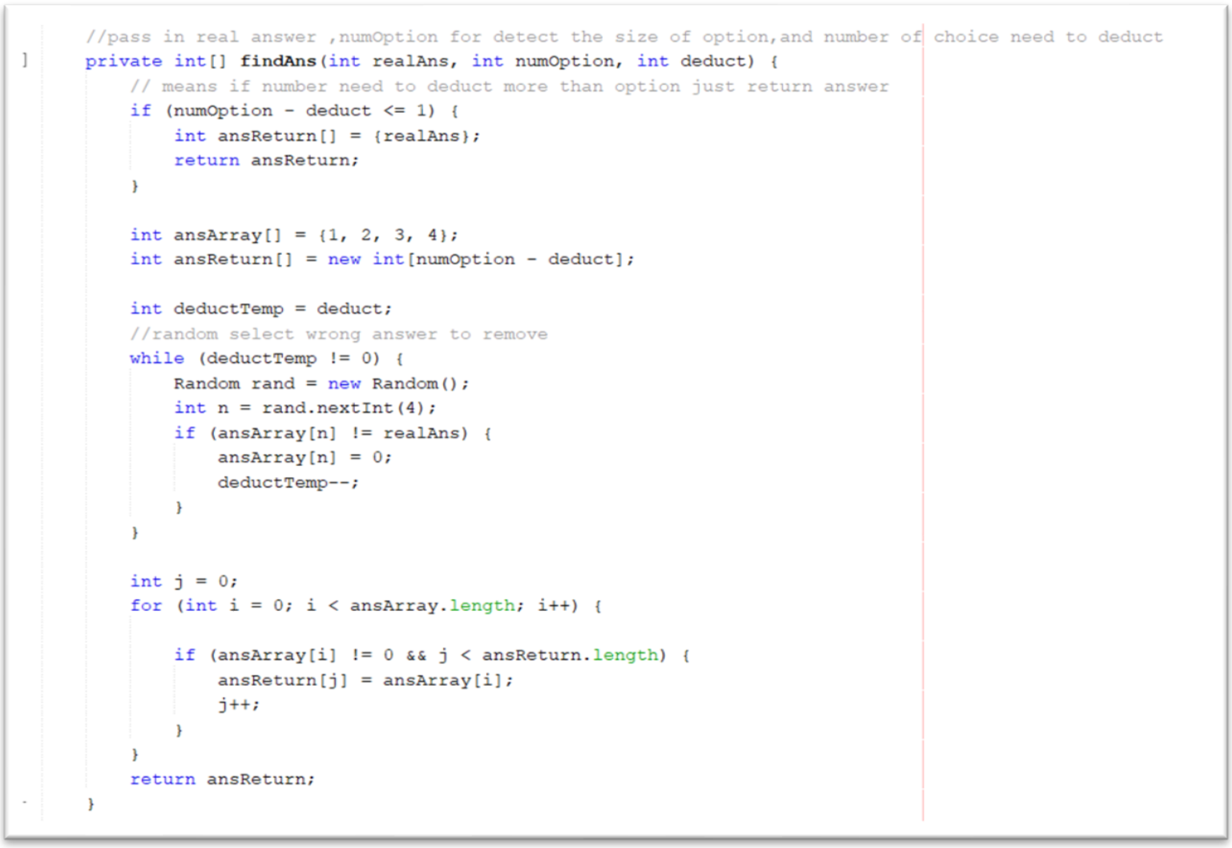


```java
} else if (nameSkill.equals("HALF")) {
    skill.deductUseTimes();
    // Do something
    int[] availableOption = findAns(rightAns, numOption, 2);
    System.out.printf("The answer you should choose are : ");
    for (int i = 0; i < availableOption.length; i++) {
        System.out.printf(" %d ", availableOption[i]);
    }

    //
    System.out.printf("\nRemaining use times : %d \n", skill.getUseTimes());
} //heal skill
else if (nameSkill.equals("HEAL")) {
    skill.deductUseTimes();
    life++;
    System.out.println("Your Life now + 1");
    System.out.printf("Remaining use times : %d \n", skill.getUseTimes());
} else if (nameSkill.compareTo("ONE") == 0) {
    skill.deductUseTimes();

    int[] availableOption = findAns(rightAns, numOption, 1);
    System.out.printf("The answer you should choose are : ");
    for (int i = 0; i < availableOption.length; i++) {
        System.out.printf(" %d ", availableOption[i]);
    }

    System.out.printf("\nRemaining use times : %d \n", skill.getUseTimes());
} //nothing
else if (nameSkill.equals("NOTHING")) {
    skill.deductUseTimes();
    System.out.println("You might more confident now, maybe?");
    System.out.printf("Remaining use times : %d \n", skill.getUseTimes());
```



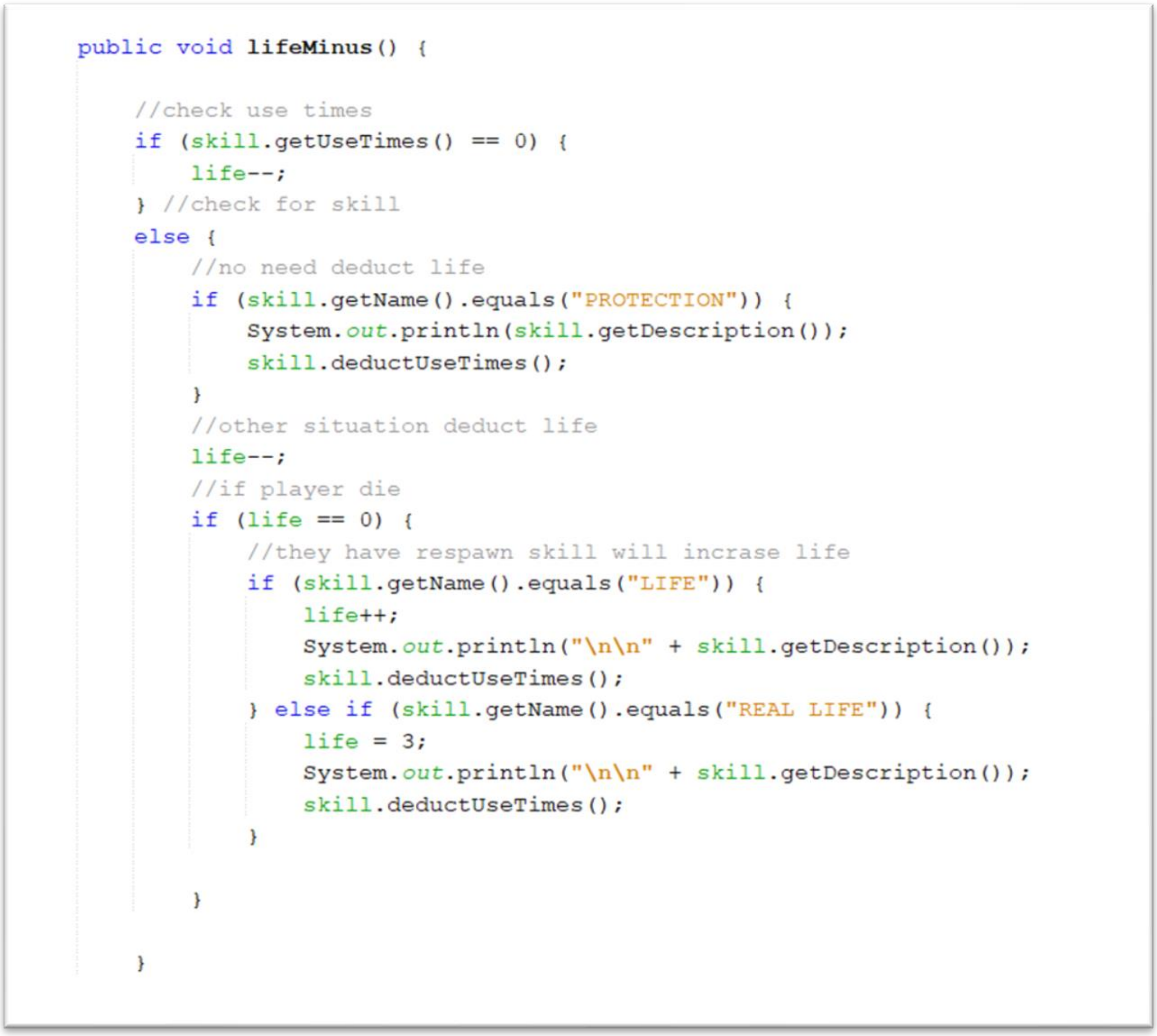
```java
            } else if (nameSkill.compareTo("ONE_ONE") == 0) {
                skill.deductUseTimes();
                int[] availableOption = findAns(rightAns, numOption, 3);
                System.out.printf("The answer you should choose are : ");
                for (int i = 0; i < availableOption.length; i++) {
                    System.out.printf(" %d ", availableOption[i]);
                }
                System.out.printf("\nRemaining use times : %d \n", skill.getUseTimes());
            } else {
                System.out.println("Invalid Skill in the code");
            }

        }

    } else {
        System.out.println("Your Skill is not available");
    }

}
```

Here is inside the Game.java. this findAns function provide the hints for player to choose the correct answer. For example, that have 4 option and player have a skill which will remove one wrong answer, this function will tell the player which three answer that should choose from the option.



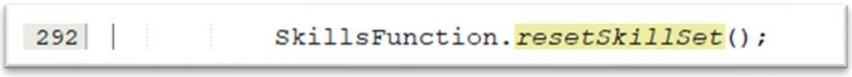
```
//pass in real answer ,numOption for detect the size of option,and number of choice need to deduct
private int[] findAns(int realAns, int numOption, int deduct) {
    // means if number need to deduct more than option just return answer
    if (numOption - deduct <= 1) {
        int ansReturn[] = {realAns};
        return ansReturn;
    }

    int ansArray[] = {1, 2, 3, 4};
    int ansReturn[] = new int[numOption - deduct];

    int deductTemp = deduct;
    //random select wrong answer to remove
    while (deductTemp != 0) {
        Random rand = new Random();
        int n = rand.nextInt(4);
        if (ansArray[n] != realAns) {
            ansArray[n] = 0;
            deductTemp--;
        }
    }

    int j = 0;
    for (int i = 0; i < ansArray.length; i++) {

        if (ansArray[i] != 0 && j < ansReturn.length) {
            ansReturn[j] = ansArray[i];
            j++;
        }
    }
    return ansReturn;
}
```

If player answer wrong will call the lifeMinus function in Game, which will deduct player's life when they don't have any protection. And also respawn the player when player run out of life and they have any relevant special skills.



```
public void lifeMinus() {

    //check use times
    if (skill.getUseTimes() == 0) {
        life--;
    } //check for skill
    else {
        //no need deduct life
        if (skill.getName().equals("PROTECTION")) {
            System.out.println(skill.getDescription());
            skill.deductUseTimes();
        }
        //other situation deduct life
        life--;
        //if player die
        if (life == 0) {
            //they have respawn skill will incrase life
            if (skill.getName().equals("LIFE")) {
                life++;
                System.out.println("\n\n" + skill.getDescription());
                skill.deductUseTimes();
            } else if (skill.getName().equals("REAL LIFE")) {
                life = 3;
                System.out.println("\n\n" + skill.getDescription());
                skill.deductUseTimes();
            }

        }

    }
```

Here is in the Driver program, this is the function will reset the skillset when the game is end.



```
            SkillsFunction.resetSkillSet();
```