UNIVERSITI MALAYA

LAB REPORT VIVA 3

CODE OF COURSE	**:**	WIX1002
COURSE NAME	**:**	FUNDAMENTALS OF PROGRAMMING
NAME OF LECTURER	**:**	DR. LIM CHEE KAU
SEMESTER & SESSION	**:**	SEMESTER 1, 2023/2024
NAME OF GROUP	**:**	OCC 10
GROUP	**:**	ZeroBaseOne

BIL.	**NAMA**	**NO. MATRIK**
1	DONG MING KANG	22121729
2	WANG YING QI	23050522
3	LONG SHUAI MING	22119548
4	XIE JIA	23054220
5	DU FENG	22100755

Viva3Q1

Author: DONG MING KANG

1.Problem Description:

The question ask us to define a class named"Eunwol" to represent some properties("location", "weapon".The class must handle properties such as location and weapon, provide methods for interaction and teleportation, and implement an interface for dimensional portal travel. The main method is provided to demonstrate the class's functionality.

2.Solution:

1.An Empty Constructor: The empty constructor initializes Eunwol with the following properties:

Weapon: "Basic Knuckles"

Location: "Grandis"

A static field named "lastLocation," also set to "Grandis"

```
1      public Eunwol(){
2          this.weapon = "Basic Knuckles";
3          this.location = "Grandis";
4          Eunwol.lastLocation = "Grandis";
5      }
```

2.A Constructor with a Location Parameter: This constructor accepts a String parameter for the character's location. It invokes the empty constructor and then updates the "location" property. It also checks if the provided location matches the static "lastLocation" and updates it accordingly.

```
1      public Eunwol(String location){
2          this();
3          setLocation();
4      }
```

3.Location Accessor and Mutator Methods: These methods handle the character's location. If a new location is different from the static "lastLocation," it updates the "location" property and updates the static "lastLocation."

```
    public void setLocation(String location){
        if(!location.equals(Eunwol.lastLocation)){
            this.location = location;
            Eunwol.lastLocation = location;
        }
    }
    public String getLocation(){
        return this.location;
    }
```

4.Weapon Accessor and Mutator Methods: These methods manage the character's weapon. The update will only occur if the new weapon contains the keyword "Knuckles" in a case-insensitive manner.

```
    public void setWeapon(String weapon){
        if(weapon.toLowerCase().contains("knuckles")){
            this.weapon = weapon;
        }
    }
    public String getWeapon(){
        return this.weapon;
    }
```

5. `toString` Method: The class should override the `toString` method to return information about the Eunwol object, including the weapon and location.

```
    public String toString(){
        return "\nEunwol Info\nWeapon:   "+getWeapon()+"\nLocation:
"+getLocation()+"\n";
```

6.Interface for Dimensional Portal: An interface named "DimensionalPortal" should be defined within the class, which includes an abstract method called "travel." This interface represents the secret arts of executing a method through a dimensional portal.

```
    public interface DimensionalPortal {
        void travel();

    }
```

3.Sample Input&Output

```
Eunwol Info
Weapon:   Genesis Knuckles
Location:     Grandis


Eunwol Info
Weapon:   Genesis Knuckles
Location:     Maple World
```

4.Source Code

```java
package viva3;
public class Eunwol {
    private String weapon;
    private String location;
    private static String lastLocation;
    public Eunwol(){
        this.weapon = "Basic Knuckles";
        this.location = "Grandis";
        Eunwol.lastLocation = "Grandis";
    }
    public Eunwol(String location){
        this();
        setLocation();
    }
    public void setLocation(String location){
        if(!location.equals(Eunwol.lastLocation)){
            this.location = location;
            Eunwol.lastLocation = location;
        }
    }
    public String getLocation(){
        return this.location;
    }
    public void setWeapon(String weapon){
        if(weapon.toLowerCase().contains("knuckles")){
            this.weapon = weapon;
        }
    }
    public String getWeapon(){
        return this.weapon;
    }
    public String toString(){
        return "\nEunwol Info\nWeapon:   "+getWeapon()+"\nLocation:
"+getLocation()+"\n";
    }
    public interface DimensionalPortal {
```

```
        void travel();

    }
}
```

```
package Viva3;
public class test {

        public static void main(String[] args) {
            Eunwol.DimensionalPortal portal = () -> {
                Eunwol eunwol = new Eunwol();
                eunwol.setWeapon("Genesis Knuckles");
                eunwol.toString();
                System.out.println(eunwol);
                eunwol.setLocation("Maple World");
                eunwol.toString();
                System.out.println(eunwol);
            };
            portal.travel();

        }
    }
```

Viva3
Question2
Author: Du Feng

2. In the eerie world of Freddy Fazbear's Pizza, the animatronic monsters have come to life, and it's your job to survive the night! You are playing as a security guard, tasked with monitoring the establishment after hours. However, something has gone terribly wrong.

   You have been asked to create a system to manage the survival equipment and the animatronic monsters' behavior. This system will be split into different classes, each with specific methods:

   `Equipment` Class to manage the flashlight and door equipment. This class should include the following methods:

   - `public Equipment[] createEquipment(Flashlight f, CloseDoor c)`: This method will initialize an array to store equipment items. It should accept a Flashlight and a CloseDoor object, add them to the array, and return the array.
   - `public void equipmentList()`: This method will display the list of equipment items with their remaining battery life (for the flashlight) and uses (for the door).

   `Flashlight` Class representing the flashlight equipment. This class should include the following method:

   - `public String getName()`: This method should return the name of the equipment, which is "Flashlight."

   `CloseDoor` Class representing the door equipment. This class should include the following method:

   - `public String getName()`: This method should return the name of the equipment, which is "CloseDoor."

   `Monster` Class to represent the different animatronic monsters. This class should include the following methods:

- `public void useUniqueAbility()`: This method should display a message describing the monster's unique ability on the character.
- `public int getBatteryReduction()`: This method should return the amount by which the monster reduces the flashlight's battery.

`Fazbear`, `Bonnie`, `Chica`, and `Foxy` Classes for each of the animatronic monsters, Fazbear, Bonnie, Chica, and Foxy. Each of these classes should include the following methods:

- `public void useUniqueAbility():` This method should override the same method in the Monster class to describe the unique ability of each animatronic.
- `public int getBatteryReduction()`: This method should override the same method in the Monster class to specify the amount by which the monster reduces the flashlight's battery.
- `public String toString()`: This method should return the name of the monster (e.g., "Fazbear," "Bonnie," "Chica," or "Foxy").

`FreddyHouse` Class to manage the monsters. This class should include the following methods:

- `public Monster[] createMonsters()`: This method will create an array to store the animatronic monsters. It should randomly generate monsters, including Fazbear, Bonnie, Chica, and Foxy, and return the array.
- `public void printMonsterAndAbilities()`: This method will display the monsters' names and their unique abilities.

`Character` Class to manage the character's equipment. This class should include the following methods:

- `public Equipment[] createEquipment(Flashlight f, CloseDoor c)`: This method should be overridden to accept a Flashlight and a CloseDoor object, add them to the character's equipment list, and return the list.

- `public void equipmentList()`: This method should be overridden to display the character's equipment list with their remaining battery life (for the flashlight) and uses (for the door).
- `public boolean useEquipment(String equipmentName, Monster monster)`: This method is optional and could be implemented to allow the character to use equipment to counter the monsters. It should accept the name of the equipment and the monster, reduce the equipment's resources (e.g., battery life or uses), and return whether the equipment was successfully used.

Main Class (`NightsOfFreddy`) where you create instances of all these classes, initialize equipment, monsters, and characters, and implement the game logic to simulate encounters between the character and animatronics.

Test Program

```
public class NightsOfFreddy
{
    public static void main(String[] args)
    {
        Flashlight fl = new Flashlight();
        CloseDoor cd = new CloseDoor();
        Character character = new Character();
        Equipment [] eqList = character.createEquipment(fl,
cd);
        character.equipmentList();

        FreddyHouse freddyHouse = new FreddyHouse(8);
        Monster [] monsList = freddyHouse.createMonsters();
        freddyHouse.printMonsterAndAbilities();

        surviveTheNight(monsList, eqList);

        if(eqList[0].getBattery() >= 0 &&
eqList[1].getDoorUses() >= 0)
        {
            System.out.println("Congrats !! You survive the
night!! ");
        }
```

```
    }
    // implement your code here
}
```

Sample Output

```
Equipment List:
Flashlight Battery: 100
CloseDoor : 10

Monster:
Bonnie can only be blocked by flashlight.
Fazbear can be blocked by flashlight or close door.
Chica can only be blocked by both flashlight and close
door.
Foxy can only be blocked by close door.
Monster List: Chica
Monster List: Bonnie
Monster List: Foxy
Monster List: Foxy
Monster List: Fazbear
Monster List: Chica
Monster List: Chica
Monster List: Bonnie

round 1: 1
flashLight: 92
close door: 9

round 1: 2
flashLight: 85
close door: 9

round 1: 3
flashLight: 85
close door: 8

round 1: 4
flashLight: 85
close door: 7
```

```
round 1: 5
flashLight: 80
close door: 7

round 1: 6
flashLight: 72
close door: 6

round 1: 7
flashLight: 64
close door: 5

round 1: 8
flashLight: 57
close door: 5

Congrats !! You survive the night!!
```

Problem solving

according to the requirement of the topic, we need to write a Java program, it can be used to manage some items and monster, because of the goods and the monster types and some properties of different deviation, we need to set up a lot of class and method to manage alone. In the above questions, the requirements for these items and monsters have been put forward in detail, and we will not repeat them here. In addition, the above problems give detailed console output and test code, which provides a lot of reference material for the writing of the program. Step1 based on the above summary, I plan to write a class according to different requirements, the following, in fact, this is a cover the superclass method to describe the unique ability of each animal

```
public class Bonnie extends Monster {
    1 个用法
    private String name = "Bonnie";

    //重写父类方法
    //这个方法应该覆盖父类方法来描述每个电子动物的独特能力
    @Override
    public void useUniqueAbility() {
        System.out.println("Bonnie can only be blocked by flashlight.");
    }
```

```
This method should override the use of the same method in the Monster class to specify the number of monsters to reduce the torch's battery
```

```
    //重写父类方法
    //这个方法应该覆盖在Monster类中使用相同的方法来指定怪物的数量减少手电筒的电池。
    @Override
    public int getBatteryReduction() {
        return 7;
    }

    //返回对象名字
    @Override
    public String toString() { return this.name; }
}
```

A total of five similar documents appear, representing five different animals, which I will not describe because their function is similar to the content of the source code

In this method, I created Flashlight and CloseDoor objects and edited their functionality and usage logic according to the requirements given in the topic, as follows

```
public class Character extends Equipment {

    private static Equipment[] equipment;

    //这个方法应该被重写以接受一个 Flashlight 和一个 CloseDoor 对象，将它们添加到角色的装备列表中，并返回该列表。
    @Override
```

```
    public Equipment[] createEquipment(Flashlight f,
CloseDoor c) {
        equipment = new Equipment[]{f, c};
        return equipment;
    }

    //这个方法应该被重写为
    //显示角色的装备列表以及他们的剩余 Flashlight 和
CloseDoor
    @Override
    public void equipmentList() {
        System.out.println("Equipment List:");
        if (equipment == null) {
            System.out.println("No equipment");
        } else {
            Equipment equipment1 = equipment[0]; //
Flashlight
            if (equipment1 != null && equipment1
instanceof Flashlight) {//equipment1!=null && 判断
equipment1 对象是否是 Flashlight 对象
                Flashlight flashlight = (Flashlight)
equipment1;
```

```
                //打印剩余 Flashlight
                System.out.println(flashlight.getName()
+ ": " + equipment1.getBattery());
            }
            Equipment equipment2 = equipment[1]; //
CloseDoor
            if (equipment2 != null && equipment2
instanceof CloseDoor) { //equipment2!= null && 判断
equipment2 是否为 CloseDoor 对象
                CloseDoor closeDoor = (CloseDoor)
equipment2;
                //打印剩余 CloseDoor
                System.out.println(closeDoor.getName()
+ ": " + equipment2.getDoorUses() + "\n");
            }
        }


    }
```

I have created a total of new classes, its main function is to cooperate with other files to complete the operation of closing the door, itself will be a variety of file stitching, as follows

```
package com.report.survivalEquipment;
```

```java
public class CloseDoor extends Equipment {
    private int count;
    private String name = "CloseDoor";

    //返回对象名称
    public String getName() {
        return this.name;
    }

    //显示有多少个 CloseDoor
    @Override
    public int getDoorUses() {
        return count;
    }

    //默认 10 个 CloseDoor 设备
    public CloseDoor() {
        this.count = 10;
    }

    //根据 count 创建对象，假设 count 为 9，就有 9 个 CloseDoor 设备
```

```
    public CloseDoor(int count) {
        this.count = count;
    }
}
```

Next, I created a class that works in conjunction with the torch and door closing logic, and is primarily responsible for managing the lifetime of the torch and the number of doors closed, as follows

```
package com.report.survivalEquipment;

//设备类
public class Equipment {

    //这个方法将初始化一个数组来存储设备项目。它应该接受一个 Flashlight 和一个 CloseDoor 对象，将它们添加到数组中，然后返回数组
    public Equipment[] createEquipment(Flashlight f, CloseDoor c) {

        return null;
    }

    //这个方法将显示设备列表
    public void equipmentList() {
```

```
    }

    public void equipmentList(Equipment[] equipment) {
        if (equipment == null) {
            System.out.println("No equipment");
        } else {
            Equipment equipment1 = equipment[0]; //
Flashlight
            if (equipment1 != null && equipment1
instanceof Flashlight) {//equipment1!=null && 判断
equipment1对象是否是 Flashlight 对象
                //打印剩余 Flashlight
                System.out.println("flashLight: " +
equipment1.getBattery());
            }
            Equipment equipment2 = equipment[1]; //
CloseDoor
            if (equipment2 != null && equipment2
instanceof CloseDoor) { //equipment2!= null && 判断
equipment2 是否为 CloseDoor 对象
                //打印剩余 CloseDoor
```

```
                System.out.println("close door: " +
equipment2.getDoorUses() + "\n");
            }
        }
    }


    //显示电池（Flashlight）剩余多少
    public int getBattery() {
        return 0;
    }

    //显示门(CloseDoor)剩余多少
    public int getDoorUses() {
        return 0;
    }
}
package com.report.survivalEquipment;

public class Flashlight extends Equipment {

    private int count;
```

```
    private String name = "Flashlight Battery";

    //返回对象名称
    public String getName() {
        return this.name;
    }

    //显示有多少个 Flashlight
    @Override
    public int getBattery() {
        return count;
    }


    //默认 100 个 Flashlight 设备
    public Flashlight() {
        this.count = 100;
    }

    //根据 count 创建对象，假设 count 为 200，就有 200 个 Flashlight 设备
    public Flashlight(int count) {
```

```
        this.count = count;
    }
}
```

When the above is done, we need to create monsters and assign different values to their data according to the topic requirements as follows

```
public FreddyHouse(int monsterCount) {
        this.monsterCount = monsterCount;
        this.monsters = new Monster[monsterCount];
    }

    public Monster[] createMonsters() {
        //使用随机数，随机生成怪物，
        //随机生成 0——3： 0 代表 Bonnie， 1 代表 Chica，2 代表 Fazbear，3 代表 Foxy
        for (int i = 0; i < monsterCount; i++) {
            int num = r.nextInt(4);
            if (num == 0) {
                monsters[i] = new Bonnie();
                monsters[i] = new Chica();
            } else if (num == 2) {
                monsters[i] = new Fazbear();
            } else if (num == 3) {
```

```
                monsters[i] = new Foxy();
            }
        }
        return this.monsters;
    }

    //这个方法将显示怪物的名字和他们独特的能力
    public void printMonsterAndAbilities() {
        //打印 Monster 信息
        System.out.println("Monster: ");
        //打印 Bonnie 独特的能力
        new Bonnie().useUniqueAbility();
        //打印 Fazbear 独特的能力
        new Fazbear().useUniqueAbility();
        //打印 Chica 独特的能力
        new Chica().useUniqueAbility();
        //打印 Foxy 独特的能力
        new Foxy().useUniqueAbility();
        //循环打印 monsters 里面的怪物名字
        for (Monster m : this.monsters) {
            System.out.println("Monster: " +
m.toString());
```

```
        }
        System.out.println("");
    }
}
```

When all the preparations are complete, we can create a main class to connect all the above and meet all the requirements of the topic, the main class content is as follows

```
package com.report;



import com.report.animatronicMonsters.*;
import com.report.survivalEquipment.Character;
import com.report.survivalEquipment.CloseDoor;
import com.report.survivalEquipment.Equipment;
import com.report.survivalEquipment.Flashlight;

public class NightsOfFreddy {



    public static void main(String[] args) {
        //给定代码
        Flashlight fl = new Flashlight();
        CloseDoor cd = new CloseDoor();
```

```
        Character character = new Character();
        Equipment[] eqList =
character.createEquipment(fl, cd);
        character.equipmentList();
        FreddyHouse freddyHouse = new FreddyHouse(8);
        Monster[] monsList =
freddyHouse.createMonsters();
        freddyHouse.printMonsterAndAbilities();
        surviveTheNight(monsList, eqList, 1);
    }


    // implement your code here
    //自己写的
    public static void surviveTheNight(Monster[]
monsters, Equipment[] equipment, int rounds) {
        Equipment flashlight;
        Equipment closeDoor;
        try {
            flashlight = equipment[0];
            closeDoor = equipment[1];
        } catch (NullPointerException
```

```
nullPointerExceptionn) {
            System.out.println("The equipment cannot
be empty");
            return;
        } catch (IndexOutOfBoundsException
indexOutOfBoundsException) {
            System.out.println("The equipment cannot
be 0");
            return;
        }
        //获取剩余 flashlight
        int battery = flashlight.getBattery();
        //获取剩余 closeDoor
        int doorUses = closeDoor.getDoorUses();
        Monster monster;
        //循环怪物
        boolean flag = true;
        for (int i = 0; i < monsters.length; i++) {
            if (flag) {
                System.out.println("round " + rounds +
": " + (i + 1));
                //获取怪物减少手电筒电池电量的数量
```

```
                monster = monsters[i];
                int batteryReduction =
monster.getBatteryReduction();
                //判断怪物类型，根据不同类型进行不同操
作
                if (monster instanceof Bonnie) {
                    if (battery >= batteryReduction) {
                        battery = battery -
batteryReduction;
                        equipment[0] = new
Flashlight(battery);
                        //显示设备列表

flashlight.equipmentList(equipment);
                    } else {
                        flag = false;
                        System.out.println("You didn't
survive the night!!");
                        break;
                    }

                } else if (monster instanceof Chica) {
```

```
                    if (battery >= batteryReduction &&
doorUses >= 1) {
                        battery = battery -
batteryReduction;
                        equipment[0] = new
Flashlight(battery);
                        equipment[1] = new CloseDoor(-
-doorUses);
                        //显示设备列表

flashlight.equipmentList(equipment);
                    } else {
                        flag = false;
                        System.out.println("You didn't
survive the night!!");
                        break;
                    }

                } else if (monster instanceof Fazbear) {
                    if (battery >= batteryReduction) {
                        battery = battery -
batteryReduction;
```

```
                        equipment[0] = new
Flashlight(battery);
                        //显示设备列表

flashlight.equipmentList(equipment);
                    } else if (doorUses >= 1) {
                        equipment[1] = new CloseDoor(-
-doorUses);
                        //显示设备列表

flashlight.equipmentList(equipment);
                    } else {
                        flag = false;
                        System.out.println("You didn't
survive the night!!");
                        break;
                    }

                } else if (monster instanceof Foxy) {
                    if (doorUses >= 1) {
                        equipment[1] = new CloseDoor(-
-doorUses);
```

```
                        //显示设备列表

flashlight.equipmentList(equipment);
                    } else {
                        flag = false;
                        System.out.println("You didn't
survive the night!!");
                        break;
                    }
                }
            }


        }
        if (flag) {
            System.out.println("Congrats !! You survive
the night !!");
        }

    }
}
```

Test Phase:

When I try to run this code, the console displays something like this

```
C:\Program Files\Java\jdk-18\bin\java.exe
Equipment List:
Flashlight Battery: 100
CloseDoor: 10

Monster:
Bonnie can only be blocked by flashlight.
Fazbear can be blocked by flashlight or close door.
Chica can only be blocked by both flashlight and close door.
Foxy can only be blocked by close door.
Monster: Fazbear
Monster: Foxy
Monster: Chica
Monster: Foxy
Monster: Fazbear
Monster: Foxy
Monster: Chica
Monster: Chica
```

```
round 1: 1
flashLight: 95
close door: 10

round 1: 2
flashLight: 95
close door: 9

round 1: 3
flashLight: 87
close door: 8

round 1: 4
flashLight: 87
close door: 7

round 1: 5
flashLight: 82
close door: 7
```

```
round 1: 5
flashLight: 82
close door: 7

round 1: 6
flashLight: 82
close door: 6

round 1: 7
flashLight: 74
close door: 5

round 1: 8
flashLight: 66
close door: 4

Congrats !! You survive the night !!
```

Source code:

Bonnie:

```java
package com.report.animatronicMonsters;

public class Bonnie extends Monster {
    private String name = "Bonnie";

    @Override
    public void useUniqueAbility() {
        System.out.println("Bonnie can only be blocked by flashlight.");
    }


    @Override
    public int getBatteryReduction() {
        return 7;
    }

    @Override
    public String toString() {
        return this.name;
    }
}
```

Chica

```java
package com.report.animatronicMonsters;

public class Chica extends Monster {
    private String name = "Chica";

    @Override
    public void useUniqueAbility() {
        System.out.println("Chica can only be blocked by both flashlight and close door.");
    }



    @Override
    public int getBatteryReduction() {
        return 8;
    @Override
```

```
    public String toString() {
        return this.name;
    }
}
```

Fazbear

```
package com.report.animatronicMonsters;

public class Fazbear extends Monster {
    private String name = "Fazbear";

    @Override
    public void useUniqueAbility() {
        System.out.println("Fazbear can be blocked by flashlight or close door.");
    }


    @Override
    public int getBatteryReduction() {
        return 5;
    }

    //返回对象名字
    @Override
    public String toString() {
        return this.name;
    }
}
```

Foxy

```
package com.report.animatronicMonsters;

public class Foxy extends Monster {
    private String name = "Foxy";

    @Override
    public void useUniqueAbility() {
        System.out.println("Foxy can only be blocked by close door.");
    }


    @Override
    public int getBatteryReduction() {
```

```
        return 0;
    }

    @Override
    public String toString() {
        return this.name;
    }
}
```

Monster

```
package com.report.animatronicMonsters;


public class Monster {

    public void useUniqueAbility(){

    }

    public int getBatteryReduction(){
        return 0;
    }
}
```

Character

```
package com.report.survivalEquipment;

import com.report.animatronicMonsters.*;
import com.report.survivalEquipment.CloseDoor;
import com.report.survivalEquipment.Equipment;
import com.report.survivalEquipment.Flashlight;

public class Character extends Equipment {

    private static Equipment[] equipment;

    @Override
    public Equipment[] createEquipment(Flashlight f, CloseDoor c) {
        equipment = new Equipment[]{f, c};
        return equipment;
    }
    @Override
    public void equipmentList() {
        System.out.println("Equipment List:");
```

```
20          if (equipment == null) {
21              System.out.println("No equipment");
22          } else {
23              Equipment equipment1 = equipment[0]; // Flashlight
24              if (equipment1 != null && equipment1 instanceof Flashlight) {
25                  Flashlight flashlight = (Flashlight) equipment1;
26                  System.out.println(flashlight.getName() + ": " +
    equipment1.getBattery());
27              }
28              Equipment equipment2 = equipment[1]; // CloseDoor
29              if (equipment2 != null && equipment2 instanceof CloseDoor) {
30                  CloseDoor closeDoor = (CloseDoor) equipment2;
31
32                  System.out.println(closeDoor.getName() + ": " +
    equipment2.getDoorUses() + "\n");
33              }
34          }
35
36      }
37
38      public boolean useEquipment(String equipmentName, Monster monster) {
39
40          String[] split = equipmentName.split(",");
41          String f = "";
42          String c = "";
43          if (split.length == 1) {
44              f = split[0];
45              c = split[0];
46          } else if (split.length == 2) {
47              f = split[0];
48              c = split[1];
49          }
50          String flashlightName = new Flashlight().getName();
51          String closeDoorName = new CloseDoor().getName();
52          if (!f.equals(flashlightName) && !c.equals(closeDoorName)) {
53              System.out.println("Please pass in the correct equipment name");
54              return false;
55          }
56          Equipment flashlight = equipment[0];
57          int battery = flashlight.getBattery();
58          Equipment closeDoor = equipment[1];
59          int doorUses = closeDoor.getDoorUses();
60          int batteryReduction = monster.getBatteryReduction();
61          if (monster instanceof Bonnie) {
62              if (battery < batteryReduction) {
63                  System.out.println("Flashlight Battery The remaining amount does
    not repel monsters");
64                  return false;
65              } else {
```

```
                equipment[0] = new Flashlight(battery - batteryReduction);
                equipmentList();
                return true;
            }
        } else if (monster instanceof Chica) {
            if (battery ≥ batteryReduction && doorUses ≥ 1) {
                equipment[0] = new Flashlight(battery - batteryReduction);
                equipment[1] = new CloseDoor(--doorUses);
                equipmentList();
                return true;
            } else {
                System.out.println("Flashlight Battery and CloseDoor The remaining amount does not repel monsters");
                return false;
            }

        } else if (monster instanceof Fazbear) {
            if (battery ≥ batteryReduction) {
                equipment[0] = new Flashlight(battery - batteryReduction);
                equipmentList();
                return true;
            } else if (doorUses ≥ 1) {
                equipment[1] = new CloseDoor(--doorUses);
                equipmentList();
                return true;
            } else {
                System.out.println("Flashlight Battery or CloseDoor The remaining amount does not repel monsters");
                return false;
            }
        } else if (monster instanceof Foxy) {
            if (doorUses ≤ 0) {
                System.out.println("CloseDoor The remaining amount does not repel monsters");
                return false;
            } else {
                equipment[1] = new CloseDoor(--doorUses);
                equipmentList();
                return true;
            }

        }
        return false;
    }
}
```

CloseDoor

```java
package com.report.survivalEquipment;

public class CloseDoor extends Equipment {
    private int count;
    private String name = "CloseDoor";

    public String getName() {
        return this.name;
    }

    @Override
    public int getDoorUses() {
        return count;
    }

    public CloseDoor() {
        this.count = 10;
    }

    public CloseDoor(int count) {
        this.count = count;
    }
}
```

Equipment

```java
package com.report.survivalEquipment;

public class Equipment {

    public Equipment[] createEquipment(Flashlight f, CloseDoor c) {

        return null;
    }

    public void equipmentList() {

    }

    public void equipmentList(Equipment[] equipment) {
        if (equipment == null) {
            System.out.println("No equipment");
        } else {
            Equipment equipment1 = equipment[0]; // Flashlight
            if (equipment1 != null && equipment1 instanceof Flashlight) {
                System.out.println("flashLight: " + equipment1.getBattery());
            }
```

```
            Equipment equipment2 = equipment[1]; // CloseDoor
            if (equipment2 != null && equipment2 instanceof CloseDoor) {

                System.out.println("close door: " + equipment2.getDoorUses() +
"\n");
            }
        }
    }


    public int getBattery() {
        return 0;
    }

    public int getDoorUses() {
        return 0;
    }
}
```

FlashLight

```
package com.report.survivalEquipment;

public class Flashlight extends Equipment {

    private int count;
    private String name = "Flashlight Battery";

    public String getName() {
        return this.name;
    }

    @Override
    public int getBattery() {
        return count;
    }


    public Flashlight() {
        this.count = 100;
    }

    public Flashlight(int count) {
        this.count = count;
    }
}
```

FreddyHouse

```java
package com.report;

import com.report.animatronicMonsters.*;

import java.util.Random;

public class FreddyHouse {
    private int monsterCount;

    private Monster[] monsters;

    private Random r = new Random();


    public FreddyHouse(int monsterCount) {
        this.monsterCount = monsterCount;
        this.monsters = new Monster[monsterCount];
    }

    public Monster[] createMonsters() {
        for (int i = 0; i < monsterCount; i++) {
            int num = r.nextInt(4);
            if (num == 0) {
                monsters[i] = new Bonnie();
            } else if (num == 1) {
                monsters[i] = new Chica();
            } else if (num == 2) {
                monsters[i] = new Fazbear();
            } else if (num == 3) {
                monsters[i] = new Foxy();
            }
        }
        return this.monsters;
    }

    public void printMonsterAndAbilities() {
        System.out.println("Monster: ");
        new Bonnie().useUniqueAbility();
        new Fazbear().useUniqueAbility();
        new Chica().useUniqueAbility();
        new Foxy().useUniqueAbility();
        for (Monster m : this.monsters) {
            System.out.println("Monster: " + m.toString());
        }
        System.out.println("");
    }
```

```
47 }
```

NightsOfFreddy

```
1  package com.report;
2
3
4  import com.report.animatronicMonsters.*;
5  import com.report.survivalEquipment.Character;
6  import com.report.survivalEquipment.CloseDoor;
7  import com.report.survivalEquipment.Equipment;
8  import com.report.survivalEquipment.Flashlight;
9
10 public class NightsOfFreddy {
11
12
13     public static void main(String[] args) {
14         Flashlight fl = new Flashlight();
15         CloseDoor cd = new CloseDoor();
16         Character character = new Character();
17         Equipment[] eqList = character.createEquipment(fl, cd);
18         character.equipmentList();
19         FreddyHouse freddyHouse = new FreddyHouse(8);
20         Monster[] monsList = freddyHouse.createMonsters();
21         freddyHouse.printMonsterAndAbilities();
22         surviveTheNight(monsList, eqList, 1);
23     }
24
25
26     // implement your code here
27     public static void surviveTheNight(Monster[] monsters, Equipment[]
   equipment, int rounds) {
28         Equipment flashlight;
29         Equipment closeDoor;
30         try {
31             flashlight = equipment[0];
32             closeDoor = equipment[1];
33         } catch (NullPointerException nullPointerExceptionn) {
34             System.out.println("The equipment cannot be empty");
35             return;
36         } catch (IndexOutOfBoundsException indexOutOfBoundsException) {
37             System.out.println("The equipment cannot be 0");
38             return;
39         }
40         int battery = flashlight.getBattery();
41         int doorUses = closeDoor.getDoorUses();
42         Monster monster;
```

```
43          boolean flag = true;
44          for (int i = 0; i < monsters.length; i++) {
45              if (flag) {
46                  System.out.println("round " + rounds + ": " + (i + 1));
47                  monster = monsters[i];
48                  int batteryReduction = monster.getBatteryReduction();
49                  if (monster instanceof Bonnie) {
50                      if (battery ≥ batteryReduction) {
51                          battery = battery - batteryReduction;
52                          equipment[0] = new Flashlight(battery);
53                          flashlight.equipmentList(equipment);
54                      } else {
55                          flag = false;
56                          System.out.println("You didn't survive the night!!");
57                          break;
58                      }
59
60                  } else if (monster instanceof Chica) {
61                      if (battery ≥ batteryReduction && doorUses ≥ 1) {
62                          battery = battery - batteryReduction;
63                          equipment[0] = new Flashlight(battery);
64                          equipment[1] = new CloseDoor(--doorUses);
65                          flashlight.equipmentList(equipment);
66                      } else {
67                          flag = false;
68                          System.out.println("You didn't survive the night!!");
69                          break;
70                      }
71
72                  } else if (monster instanceof Fazbear) {
73                      if (battery ≥ batteryReduction) {
74                          battery = battery - batteryReduction;
75                          equipment[0] = new Flashlight(battery);
76                          flashlight.equipmentList(equipment);
77                      } else if (doorUses ≥ 1) {
78                          equipment[1] = new CloseDoor(--doorUses);
79                          flashlight.equipmentList(equipment);
80                      } else {
81                          flag = false;
82                          System.out.println("You didn't survive the night!!");
83                          break;
84                      }
85
86                  } else if (monster instanceof Foxy) {
87                      if (doorUses ≥ 1) {
88                          equipment[1] = new CloseDoor(--doorUses);
89                          flashlight.equipmentList(equipment);
90                      } else {
91                          flag = false;
```

```
92                          System.out.println("You didn't survive the night!!");
93                          break;
94                      }
95                  }
96              }
97
98
99          }
100         if (flag) {
101             System.out.println("Congrats !! You survive the night !!");
102         }
103
104     }
105 }
```