

Forum:[Hack Forums](#)

Thread: [Slot Machine](#)

Nemo2478 - October 26, 2019, 2:18 am

Hello everybody,

slot machine exercise that has been done for college.

Source Code: [Slot Machine - Google Drive](#)

Regards

MavenGradle - October 26, 2019, 8:37 am

Holy shit [Nemo2478](#) , You're still using JDK 1.7? Allow me to introduce you to 2019:

<https://adoptopenjdk.net/?variant=openjdk13&jvmVariant=hotspot>

Next up. Your code. You produced something that is actually a lot better than your previous releases, regardless of your usage of old JDK versions. Better, but not fantastic or good at all. There are quite a few remarks that I can make to hopefully allow you to improve.

First of all, your Main class:

```
Code System.out.println("Enter the credit: ");
```

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

```
double credit = keyboard.nextDouble();
```

No exception handling. I can simply enter any NaN value and your application will crash because you're not checking the input.

I can completely stop your entire game simply by entering a negative credit to begin with, which is something you'll probably want to restrict.

I can also break your application by entering a credit value that is smaller than the cost of 'playing'. You should check for this:

```
Code Machine machine = new Machine(3, 0.5, award1, award2);Code if (getCredit() >= gambPri) {
    avaiCred = avaiCred - gambPri;
```

```
Random generator = new Random();
```

```
Fruit[] fruits;
```

```
fruits = new Fruit[boxes];
```

```
Fruit[] random_fruits=Fruit.values();
```

```
for (int i = 0;i<boxes; i++){
    int index = generator.nextInt(5);
    fruits[i]=random_fruits[index];
}
```

```
int n = 1; // n = Number of prizes
```

```
for (int i = 0; i <= n; i++) {
```

```
// check if combination is in registered prizes
```

```
if (Arrays.equals(fruits, collection[i].getCombStock()) == true){
```

```

    avaiCred = avaiCred + collection[i].getAward();
}
}
return fruits;
}

```

return null; This causes your function to return null instead of a proper exception or simply not allowing plays when credit < the amount required to play. Right now I'm stuck in an infinite loop. Thanks for the infinite loop, Nemo..

This is also completely redundant, I don't know why you keep doing this in your play method:

```

Code public Fruit[] play() {
    setGambPri(gambPri);
    ...
}

```

You do not have to keep setting the 'gamb pri' whatever the hell that even means. Doing it once is more than enough and it serves absolutely no purpose at all to set a value to be equal to itself over and over again when playing. You already set this value in the constructor:

```

Code public Machine(int nBoxes, double price, Award... award) {
    gambPri=price;
    boxes=nBoxes;
    collection=award;
}

```

But your constructor does not properly handle illegal variable values. You can easily make a Machine with a negative 'gambPri' (fuck man, use proper variable names) for example.

This method right here returns a value yet you never use this value:

```

Code public double increaseCredit(double increase) {
    return avaiCred + increase;
}

```

You also only use this once at the beginning of the 'game', which makes this function entirely redundant. It serves absolutely no purpose. It doesn't do anything. You have another function that does set the credit correctly if that is what you intended to do, but for some reason you named it poorly (like you so often do):

```

Code public void setAvaiCred(double credit) {
    avaiCred = credit;
}

```

In addition to the naming of this function, you should not allow negative credit to be passed.

In your Award class, your constructor is supposed to add a credit to the total balance of the user if he wins, yet you can easily put negative numbers and null arrays in here since you once again do not check the values of the variables that are passed. This can cause behaviour that you do not want for the purpose of this 'game':

```

Code public Award(Fruit[] combination, int a) {
    CombStock=combination;
    award=a;
}

```

You also have quite a few methods in there that serve absolutely no purpose because they're not used anywhere:

```

Code public double collect() {
    System.out.println("Removing "+avaiCred+" dollars from the machine");
    avaiCred = 0;
    return avaiCred;
}

public int getnBoxes() {
    return boxes;
}

```

```
public Award[] getColeccion() {  
    return collection;  
}  
public double getGambPri() {  
    return gambPri;  
}
```

Right here, you do something that is actually quite funny:

```
Code if (Arrays.equals(fruits, collection[i].getCombStock()) == true){
```

```
...
```

```
}You can simplify this:
```

```
Code if (Arrays.equals(fruits, collection[i].getCombStock())){
```

```
...
```

```
}
```

By the way, you also do not close your Scanner. Surely you must know why I tell you this by now..

So to conclude:Use proper variable names

Use proper getters and setters

Use proper function names

Use proper error handling

Restrict user input to values that make sense

Remove unused functions

Close your fucking scanner

Stop using JDK 1.7

Thank you Nemo, for correcting these flaws (which I'm sure you won't, as usual).