Name : Ayat Jamhour **Part1 / part 2 : OOP concept / the snack machine UML diagram**

23/oct/2021

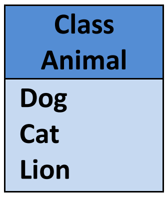
Part 1 :

it will add a lot , it will show me my weaknesses and help me to improve them
It makes me more confident in myself , and learn about new questions and problems that may encounter and how to solve them .

Part 2 :

OOP (object oriented program) : "objects and classes"

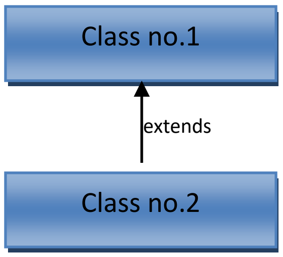
Use it to organize the code , make it easy to read and maintenance , clear structure and make it a reusable code "less and short ".



Main concepts :

1- **Inheritance .**
2- **Polymorphism .**
3- **Abstraction .**
4- **Encapsulation .**

- **Inheritance** : inherit all thing like attribute or behavior from class to another . It have a **superClass** "Parent" and **subclass** "child" , we use a keyword "**extends** " to inherit from class .



```
class Animal {

public void eat() {

}}

class Dog extends Animal {

// new method

}
```

- **Polymorphism** : many forms or shapes , here we have some classes related to each other by "**inheritance**" , It have to concept "**overriding**" and "**overloading**", polymorphism use the method and attribute that has been inherited and use it to different and new task .

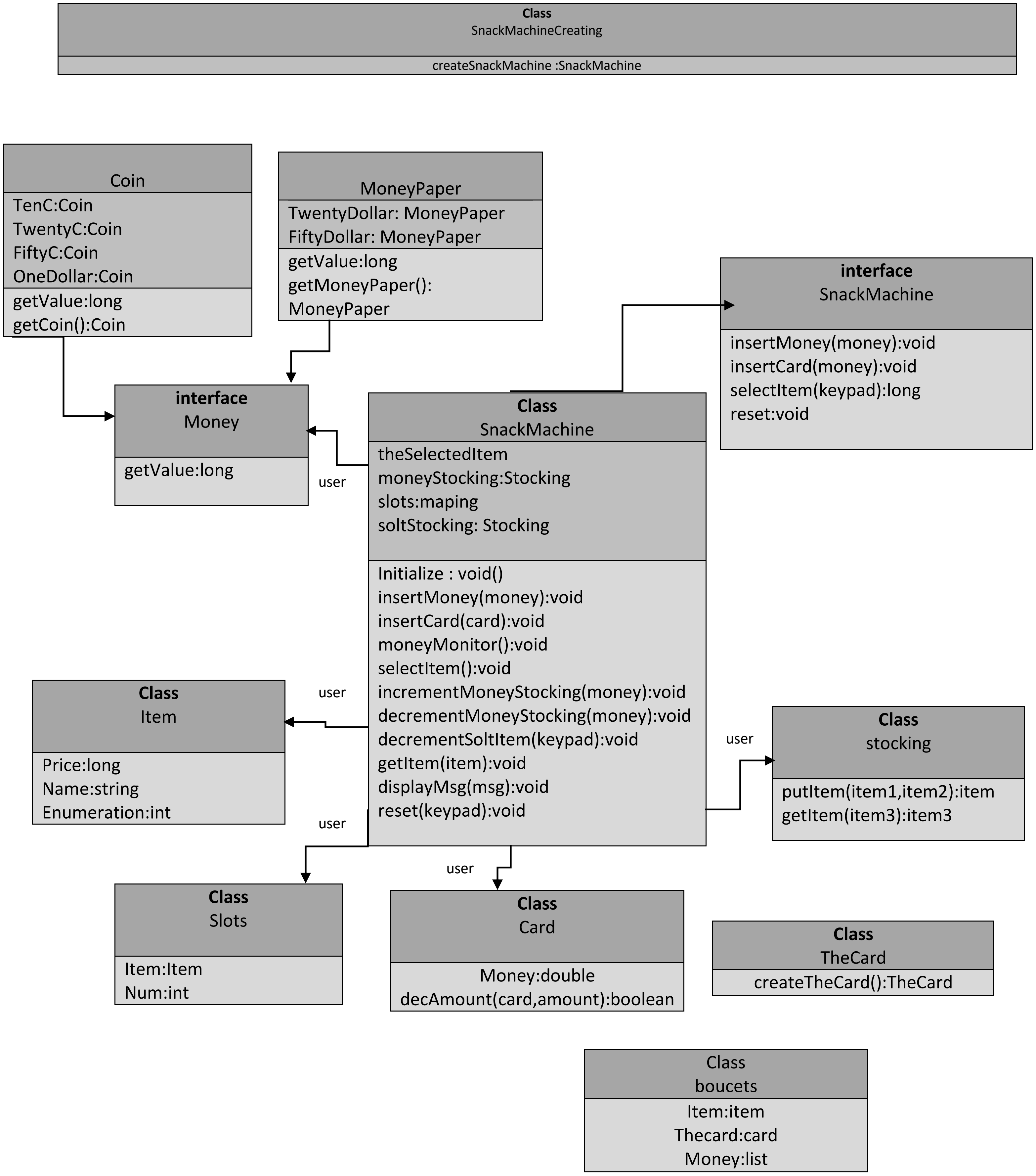
```
class Animal {

public void fourLegs (){

system.out.print("four legs");}

}

class Cat extends Animal {

public void fourLegs (){

system.out.print("cat has four legs");}}

class Dog extends Animal {

public void fourLegs (){

system.out.print("dog has four legs");}}
```

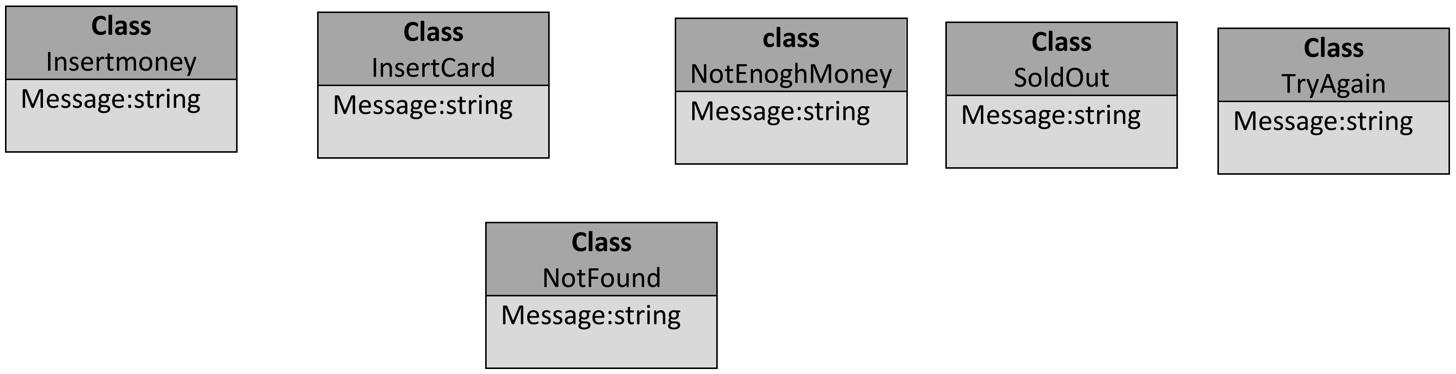
- **Abstraction** : "for security" it will be hidden the certain data from user , it can be achieved with class , method or interface , the "**abstract method** " doesn't have a body , and to access the "**abstract class**" , should be use the inheritance .

```
abstract class Animal {

public abstract void fourLegs ();

public  void eat (){

system.out.print("eat");}}

class Cat extends Animal {

public void fourLegs (){

system.out.print("cat has four legs");}}

class Main {

public static void main (String[] args){

Cat newCat = new Cat ();

newCat.fourLegs();

newCat.eat();}}
```

- <u>**Encabsulation**</u> : “for security” hidden sensitive data from user ,change one part of the code without effecting to another one , it use the “**private**” make the class or attributes as a “**private**” , to access and update them it use “**get :read only** ” and “**set : write only** ” . Use get to return and get the private variable and set to set a new value

```
public class  Student  {

private string name ;

private Integer age ;

public String getName (){

return name;}}

public void setName (String name){{

this.name  = name ;

}}

public String getAge (){

return name;}}

public void setAge (Integer  Age){{

this.age  = age ;

}}
```


Class
SnackMachineCreating
createSnackMachine :SnackMachine
Coin
TenC:Coin
TwentyC:Coin
FiftyC:Coin
OneDollar:Coin
getValue:long
getCoin():Coin
MoneyPaper
TwentyDollar: MoneyPaper
FiftyDollar: MoneyPaper
getValue:long
getMoneyPaper():
MoneyPaper
interface
SnackMachine
insertMoney(money):void
insertCard(money):void
selectItem(keypad):long
reset:void
interface
Money
getValue:long
user
Class
SnackMachine
theSelectedItem
moneyStocking:Stocking
slots:maping
soltStocking: Stocking
Initialize : void()
insertMoney(money):void
insertCard(card):void
moneyMonitor():void
selectItem():void
incrementMoneyStocking(money):void
decrementMoneyStocking(money):void
decrementSoltItem(keypad):void
getItem(item):void
displayMsg(msg):void
reset(keypad):void
user
Class
Item
Price:long
Name:string
Enumeration:int
user
Class
stocking
putItem(item1,item2):item
getItem(item3):item3
user
user
Class
Slots
Item:Item
Num:int
Class
Card
Money:double
decAmount(card,amount):boolean
Class
TheCard
createTheCard():TheCard
Class
boucets
Item:item
Thecard:card
Money:list


Class
Insertmoney
Message:string
Class
InsertCard
Message:string
class
NotEnoghMoney
Message:string
Class
SoldOut
Message:string
Class
TryAgain
Message:string
Class
NotFound
Message:string