Modélisation et implémentation d'un Pokédeck

TP 3-4

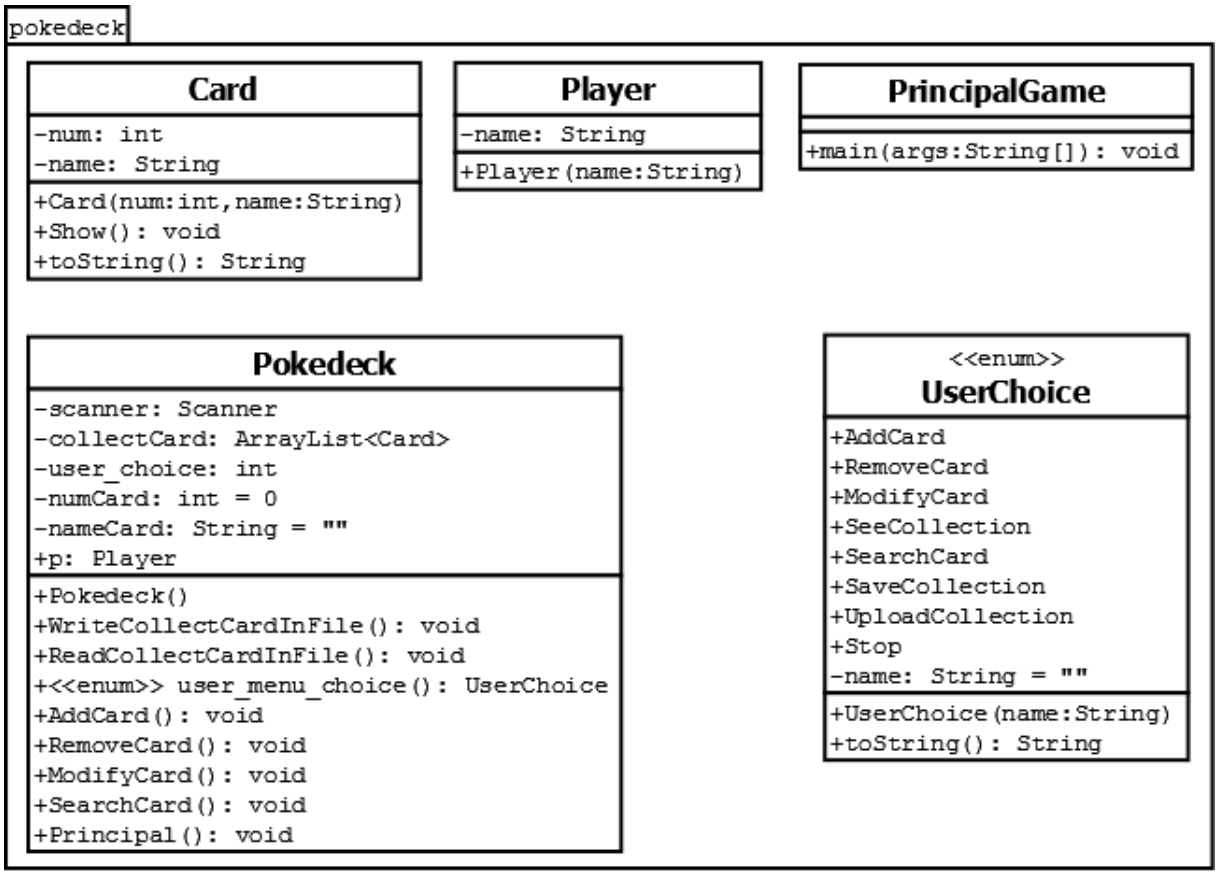
Programmation objets, web et mobiles en JAVA
Licence 3 Professionnelle – Multimédia

WALTHER Laureen

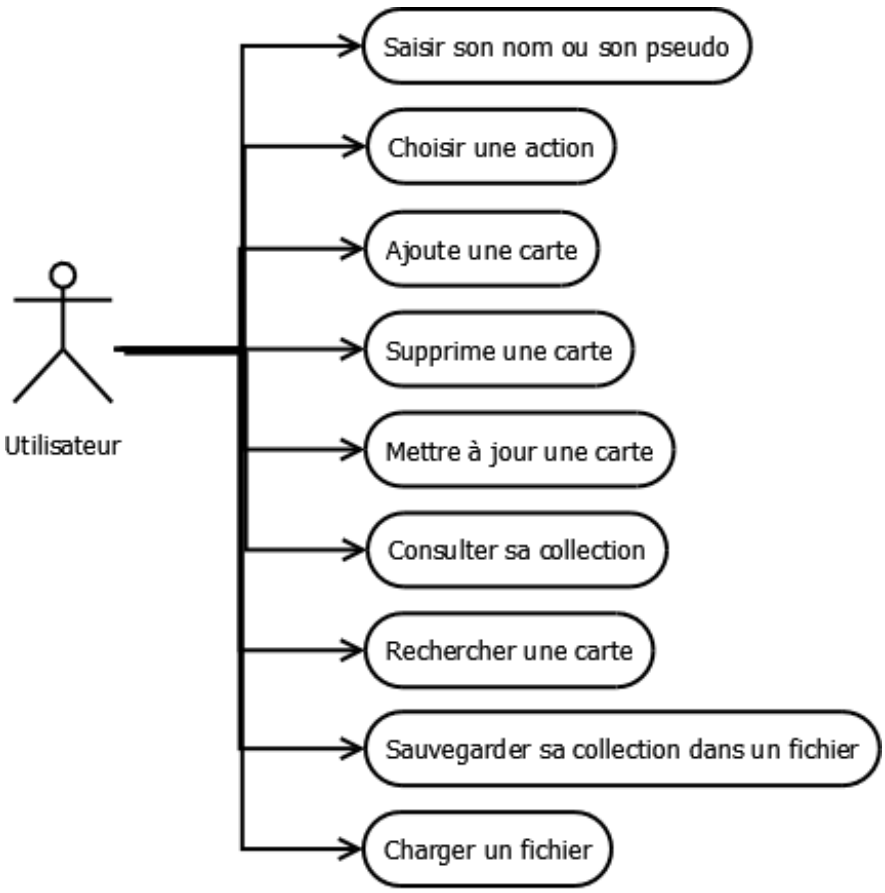
Université Pierre et Marie Curie

The aim of this work was to implement a Pokedeck, that is to say, software to manage their Pokemon cards.

1) Class diagram (UML)



2) Use case



3) Class description

1. **Card**
   - Contains constructor to create card with parameters num and name
   - **Attributes :** num (in type int) et name (in type String)
   - **Methods :**
     - Show() : to display card
     - toString() : return string used to describe object
2. **Player :** contains player name
   - Contains constructor with parameter **name** (in type String)
   - **Attributes:** name (in type String)
   - **Accesseur** getName()
3. **Pokedeck:** contains flow of the game
   - **Attributes:**
     - collectCard (in type ArrayList<Card>) : initialize tab of cards
     - user_choice (in type int) : user choice
     - numCard (in type int) : initialize num card to zero
     - nameCard (in type String) : initialize name card to blank
   - **Objects:**
     - Scanner scanner : used to read keyboard input
     - Player p : contains the current player
   - **Methods:**
     - **WriteCollectCardInFile()**
       - FileOutputStream file: creates a file output stream to write
       - ObjectOutputStream oos**:** open stream on file
       - writeObject(): object serialization
       - flush(): empty the writing buffers
       - close(): close stream
     - **ReadCollectCardInFile()**
       - FileInputStream file: creates a file input stream to read
       - ObjectInputStream**:** open stream on file
       - readObject(): object deserialization
     - **UserChoice user_menu_choice():** Use enumeration UserChoice to propose menu at user
     - **AddCard():** add description of new card
       - **Test:** as the collection contains card name selected by user, program asks card name
       - **Test:** if the collection contains card number, add 10 at the card number
       - Add new card on the collectCard, increments card number, display collectCard
     - **RemoveCard():** delete card
       - Request the number card to remove, remove corresponding card on the collectCard

- **ModifyCard():** update card
  - Request the number card to update, request new card name, replace corresponding card with new card name
- **SeeCollection():** display collectCard
- **SearchCard():** search card
  - Request the number card to search, request the name card to search
  - **Test:** if collectCard contains number card and name card : display corresponding card
- **Principal():** launch methods
  - Request the user name
  - Display menu, as choice is different from 8, menu is displayed
  - User choice:
    - [1] : AddCard(), WriteCollectCardInFile() and display menu
    - [2] : RemoveCard(), WriteCollectCardInFile() and display menu
    - [3] : ModifyCard(), WriteCollectCardInFile() and display menu
    - [4] : ReadCollectCardInFile(), print collectCard and display menu
    - [5] : SearchCard() and display menu
    - [6] : WriteCollectCardInFile() and system exit
    - [7] : ReadCollectCardInFile() and display menu
    - [8] : System exit

4) Flow of the game

a. Basics

This software would allow the user to make two basic actions:

[1] Add description of new card

Code:

```
public static void AddCard() {
	do {
		System.out.println("Card name :");
		nameCard = scanner.next();
		scanner.nextLine();
	} while (collectCard.toString().contains(nameCard));
	if (collectCard.toString().contains(String.valueOf(numCard))) {
		numCard+=10;
	}
	collectCard.add(new Card(numCard, nameCard));
	numCard++;
	for (int i = 0; i < collectCard.size(); i++) {
		Card maCarte = collectCard.get(i);
		System.out.println(maCarte);
	}
}
```

Test in console:

```
[1] Add new card
[2] Delete card
1
Card name :
toto
0 toto
```

[2] Delete card

Code:

```
public static void RemoveCard() {
	System.out.println("Enter card number you want to delete : ");
	numCard = scanner.nextInt();
	Object cardDelete = collectCard.remove(numCard);
	System.out.println(cardDelete + " has been removed");
}
```

Test in console:

```
0 toto
[1] Add new card
[2] Delete card
2
Enter card number you want to delete :
0
0 toto has been removed
```

b. Extension

The program was to offer three new features to the user:

[1] Update card

Code:

```
public static void ModifyCard() {
	System.out.println("Enter card number you want to update : ");
	numCard = scanner.nextInt();
	System.out.println("New card name :");
	nameCard = scanner.next();
	scanner.nextLine();
	Object cardUpdate = collectCard.set(numCard, new Card(numCard, nameCard));
	System.out.println(cardUpdate + " has been updated");
}
```

Test in console:

```
[4] See collection
[5] Search card
[6] Save collection
[7] Upload collection
[8] Exit
3
Enter card number you want to update :
1
New card name :
tutu
1 titi has been updated
```

[2] See collection

Code :

```
ReadCollectCardInFile();
System.out.println("Collection : "+collectCard);
```

Test in console:

```
1 titi
[1] Add new card
[2] Delete card
[3] Update card
[4] See collection
[5] Search card
[6] Save collection
[7] Upload collection
[8] Exit
4
Collection : [0 toto, 1 titi]
```

[3] Search card

```
public static void SearchCard() {
	System.out.println("Enter card number you want to search :");
	int numCardSearch = scanner.nextInt();
	System.out.println("Enter card name you want to search : ");
	String nameCardSearch = scanner.next();
	scanner.nextLine();
	if (collectCard.toString().contains(new Card(numCardSearch,
nameCardSearch).toString())) {
		System.out.println("Your card : "+new Card(numCardSearch,
	nameCardSearch).toString());
	} else {
		System.out.println("Your collection does not contain card : "+new
Card(numCardSearch, nameCardSearch).toString());
	}
}
```

```
0
Enter card name you want to search :
titi
Your collection does not contain card : 0 titi
[1] Add new card
[2] Delete card
[3] Update card
[4] See collection
[5] Search card
[6] Save collection
```

5) Save collection in file

```
public static void WriteCollectCardInFile() {
	try {
		FileOutputStream file = new FileOutputStream(p.getName()+".txt");
		ObjectOutputStream oos = new ObjectOutputStream(file);
		oos.writeObject(collectCard);
		oos.flush();
		oos.close();
	} catch (IOException e) {
		e.printStackTrace();
	}
}

public class Card implements Serializable
```

```
[2] Delete card
[3] Update card
[4] See collection
[5] Search card
[6] Save collection
[7] Upload collection
[8] Exit
6
Backup file : test.txt
```

6) Upload file

```
public static void ReadCollectCardInFile() {
	try {
		FileInputStream file = new FileInputStream(p.getName()+".txt");
		ObjectInputStream ois = new ObjectInputStream(file);
		collectCard = (ArrayList<Card>) ois.readObject();
	} catch (java.io.IOException e) {
		e.printStackTrace();
	} catch (ClassNotFoundException e) {
		e.printStackTrace();
	}
}
```

```
[1] Add new card
[2] Delete card
[3] Update card
[4] See collection
[5] Search card
[6] Save collection
[7] Upload collection
[8] Exit
4
Collection : [0 toto, 1 titi, 2 laureen]
```

7) Menu

Code :

```
public enum UserChoice {
	AddCard ("Add new card"),
	RemoveCard ("Delete card"),
	ModifyCard ("Update card"),
	SeeCollection ("See collection"),
	SearchCard ("Search card"),
	SaveCollection ("Save collection"),
	UploadCollection ("Upload collection"),
	Stop ("Exit");

	private String name = "";

	UserChoice(String name) {
		this.name = name;
	}

	public String toString() {
		return name;
	}
}

public static UserChoice user_menu_choice() {
	System.out.println("[1] "+UserChoice.AddCard+"\n"
			+ "[2] "+UserChoice.RemoveCard+"\n"
			+ "[3] "+UserChoice.ModifyCard+"\n"
			+ "[4] "+UserChoice.SeeCollection+"\n"
			+ "[5] "+UserChoice.SearchCard+"\n"
			+ "[6] "+UserChoice.SaveCollection+"\n"
			+ "[7] "+UserChoice.UploadCollection+"\n"
			+ "[8] "+UserChoice.Stop);
```

```
        user_choice = scanner.nextInt();
        scanner.nextLine();
        return UserChoice.values()[user_choice-1];
}
```

8) Decoupling user interface and business classes

I am part of the existing code, only Pokedeck class was modified and PokedeckUI class has been added.

a. Pokedeck UI

Contains menus and questions, which is seen by the user.

```
public void start() {
        boolean stop = false;
        System.out.println("Enter your name : ");
        String playerName = scanner.next();
        p = new Player(playerName);
        pokedeck.setP(p);
        while (!stop) {
                UserChoice choice = user_menu_choice();
                stop = pick_choice(choice);
        }
}

private boolean pick_choice(UserChoice option) {
        boolean quit = false;
        switch (option) {
        case AddCard:
                do {
                        System.out.println("Card name :");
                        nameCard = scanner.next();
                        scanner.nextLine();
                        pokedeck.setNameCard(nameCard);
                } while (pokedeck.getCollectCard().toString().contains(nameCard));
                pokedeck.addCard();
                System.out.println(pokedeck.getMyCard());
                pokedeck.writeCollectCardInFile();
                break;
        case RemoveCard:
                System.out.println("Enter card number you want to delete : ");
                numCard = scanner.nextInt();
                pokedeck.setNumCard(numCard);
                pokedeck.removeCard();
                System.out.println(pokedeck.getCardDelete() + " has been removed");
                pokedeck.writeCollectCardInFile();
                break;
        case ModifyCard:
                System.out.println("Enter card number you want to update : ");
                numCard = scanner.nextInt();
                pokedeck.setNumCard(numCard);
                System.out.println("New card name :");
                nameCard = scanner.next();
                scanner.nextLine();
                pokedeck.setNameCard(nameCard);
                pokedeck.modifyCard();
                System.out.println(pokedeck.getCardUpdate() + " has been updated");
```

```
                pokedeck.writeCollectCardInFile();
                break;
        case SeeCollection:
                pokedeck.readCollectCardInFile();
                System.out.println("Collection : "+pokedeck.getCollectCard());
                break;
        […]
        case Stop:
                quit = true;
                break;
        default:
                System.out.println("We didn't understand your choice");
                break;
        }
        return quit;
}
```

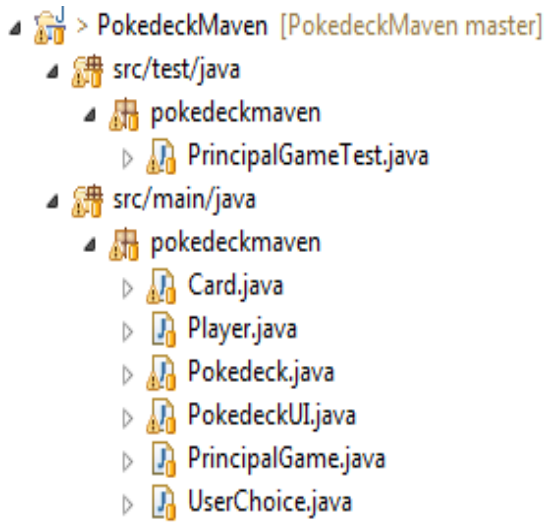
b. Pokedeck

Contains different methods, we will not find Scanner, or System.out.print.

```
public static void addCard() {
        numCard = 1 + random.nextInt(1000 - 0);
        collectCard.add(new Card(nameCard, numCard));
        for (int i = 0; i < collectCard.size(); i++) {
                myCard = collectCard.get(i);
        }
}

public static void removeCard() {
        for (int i = 0; i < collectCard.size(); i++) {
        if (collectCard.get(i).toString().contains(Integer.toString(numCard))) {
                cardDelete = collectCard.remove(i);
        }
        }
}

public static void modifyCard() {
        for (int i = 0; i < collectCard.size(); i++) {
        if (collectCard.get(i).toString().contains(Integer.toString(numCard))) {
                cardUpdate = collectCard.set(i, new Card(nameCard, numCard));
        }
        }
}

public static boolean searchCard() {
        if (collectCard.toString().contains(new Card(nameCardSearch,
numCardSearch).toString())) {
                return true;
        } else {
                return false;
        }
}
```

9) Bonus

a. Project architecture with Maven

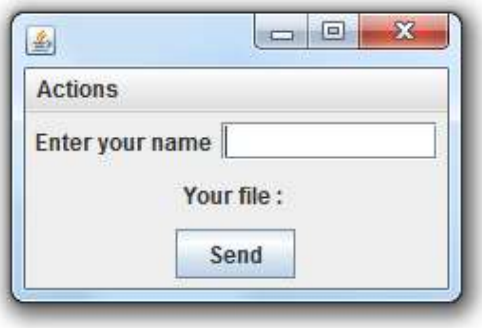


XML file : pom.xml

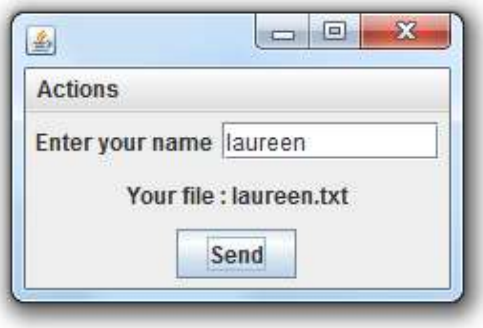
```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>pokedeckmaven</groupId>
  <artifactId>PokedeckMaven</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>PokedeckMaven</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
  <plugins>
   <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>3.1</version>
    <configuration>
     <source>1.7</source>
     <target>1.7</target>
    </configuration>
   </plugin>
  </plugins>
 </build>
</project>
```

1) In addition : Pokedeck with Graphic User Interface

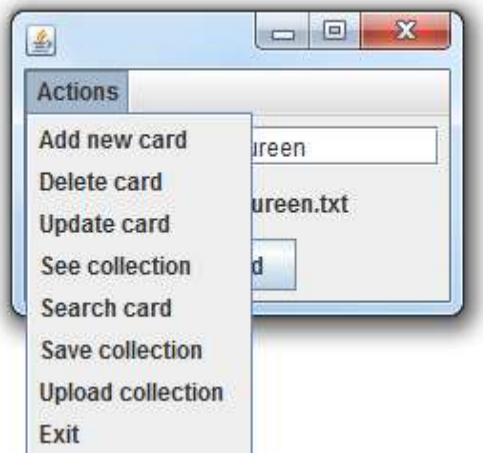
Github : https://github.com/laureenw/Pokedeck_withGUI



First interface



The user has entered his name and the program tells him that the file belongs to

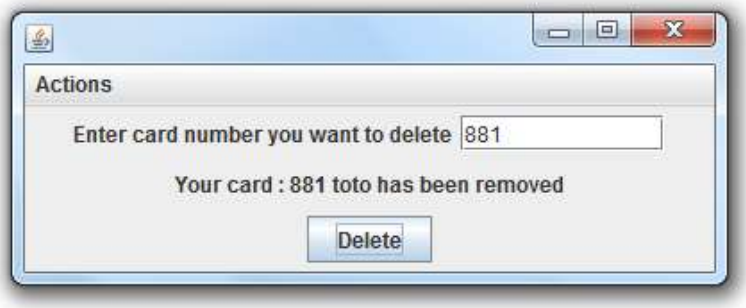


Menu

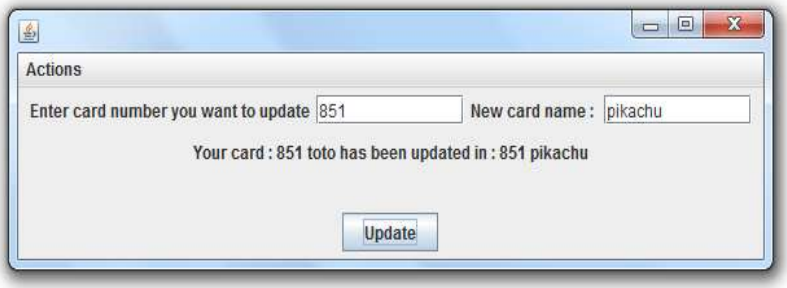
Add new card

The user has entered the name of a card and the program displays the card

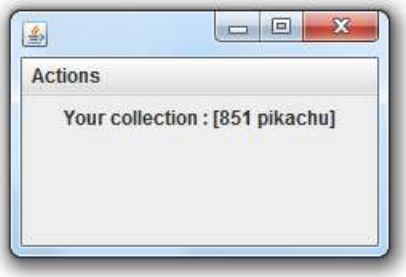
If the card name already exists in the collection



Delete card: if the card number exists in the collection, the card is removed



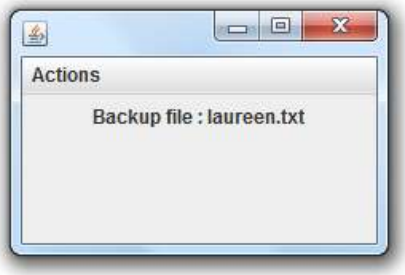
Update card: enter the card number and the new name



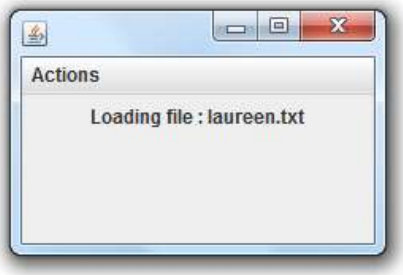
See collection

Search card: if the card does not exist in the collection

If card is in the collection



Save collection



Upload collection