

	<b>Java</b> <b>Blackjack</b>	BTS
		SNIR

## Java

### Découverte

Nous allons utiliser l'IDE Eclipse pour développer.

### Installation

<https://www.eclipse.org/downloads/packages/downloads/>

Laissez les options par défaut pour Eclipse IDE pour Java. Le premier téléchargement vous permettra d'installer l'installateur (oui) de Eclipse IDE pour Java.

Une fois l'installation terminée, lancez Eclipse. Créez un nouveau Projet de type Java, nommé « test-project ».

Laissez les options par défaut. Si Eclipse vous demande de créer un nouveau module, ignorez la question (répondez « non »).

### Classes

En Java, tout est classes. Chaque classe est contenue dans son propre fichier. Nous allons donc faire attention au nommage de nos classes, et respecter les règles suivantes :

- Utilisation du CamelCase
- Pas de \_ ou de -
- Anglais, pas d'accents

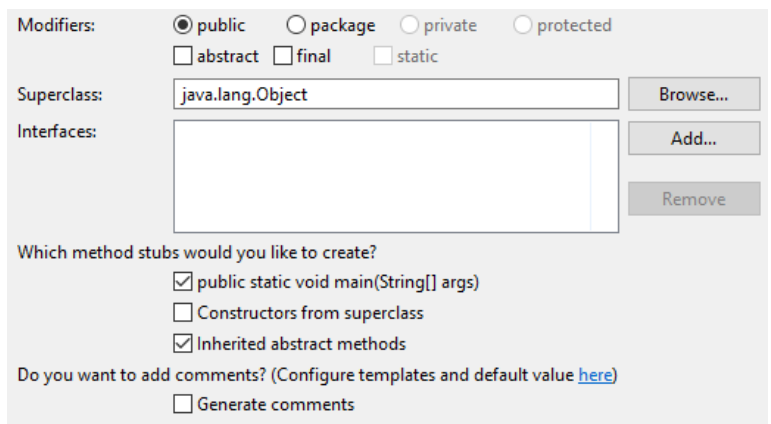
### Démarrage

Lorsque vous lancez un projet Java (en utilisant Run), celui-ci va chercher (sauf si indiqué) la classe contenant la méthode spéciale main :

```
public static void main(String[] args) {
```

Créez donc une première classe dans votre projet, et nommez-la comme votre projet.

1. En suivant les règles de nommages, quel devrait être le nom de cette classe ?



*Pensez à bien cocher la génération d'une méthode main(), comme indiqué ci-contre.*

	<p><b>Java</b></p>	<p><b>BTS</b></p>
<p><b>TP</b></p>	<p><b>Blackjack</b></p>	<p><b>SNIR</b></p>

## Affichage et chaînes

En Java, l'instruction pour afficher du texte dans la sortie (console) est la suivante :

```
System.out.println("Hello World !");
```

La méthode println() permet d'afficher du texte en sautant une ligne, tandis que la méthode print() permet d'afficher du texte sans sauter de ligne.

En Java, on peut déclarer une variable sans affecter une quelconque valeur.

La déclaration de variable se fait en spécifiant le type de variable et son nom. L'affectation se fait en spécifiant son nom suivi du symbole égal et de la valeur souhaitée.

```
String color;
color = "red";
```

```
int howOld = 15;
double size = 175.0;
float weight = 251f;
String color;
Scanner scanner = new Scanner(System.in);
boolean isTrue = false;
Boolean isAlsoTrue = false;
Object o = null;
```

Déclarez une variable « name » destinée à stocker un nom. Pour le moment, n'initialisez aucune valeur.

### 2. Affichez le contenu de la variable name. Que se passe-t-il ?

Initialisez la variable « name » à la valeur spéciale null (comme l'Object o dans l'exemple plus haut).

### 3. Que remarquez-vous ?

On peut obtenir la longueur d'une chaîne de caractères en utilisant la méthode « length() » comme ci-contre.

```
String tu = "perlimpimpin";
int len = tu.length();
```

Affichez la longueur de la variable « name ».

### 4. Qu'obtenez-vous ?

	<b>Java</b>	<b>BTS</b>
<b>TP</b>	<b>Blackjack</b>	<b>SNIR</b>

### Flux système

Le code ci-contre vous permet d'afficher la classe des trois propriétés de flux principales de System : out, in et err.

```
System.out.println(System.out.getClass());
System.out.println(System.in.getClass());
System.out.println(System.err.getClass());
System.err.println("Hello error");
```

5. Exécutez ce code et notez les résultats.

6. D'après vous, à quoi sert le flux System.out ?

7. D'après vous, à quoi sert le flux System.err ?

8. D'après vous, à quoi sert le flux System.in ?

**Vérifiez votre avancée auprès de votre enseignant.**

La classe « Scanner » permet de lire un flux, y compris la ligne de commande. Son constructeur prend en argument le flux concerné.

```
Scanner myScanner = new Scanner(myStream);
String whatTheUserTypedIn = myScanner.nextLine();
```

Une fois un objet de la classe Scanner instancié, on peut l'utiliser pour lire la ligne de commande, en utilisant diverses méthodes, comme nextLine() qui met en pause le programme, attend pour un saut de ligne, et enregistre tout ce qui a été tapé dans la ligne de commande avant le saut de ligne.

*Attention : Scanner étant une classe externe à notre fichier, nous aurons besoin de l'importer. Cela se fait en utilisant la commande import avant la définition de la classe dans notre fichier. Généralement, votre IDE vous propose de le faire pour vous.*

```
import java.util.Scanner;
```

9. Faites le nécessaire pour que votre programme demande à l'utilisateur son prénom, puis qu'il affiche « Hello » suivi du prénom qui a été saisi en ligne de commande.

10. Modifiez le code de votre classe pour demander un nom tant qu'une chaîne vide n'ai été entrée.

	<p align="center"><b>Java</b></p>	<p align="center"><b>BTS</b></p>
<p align="center"><b>TP</b></p>	<p align="center"><b>Blackjack</b></p>	<p align="center"><b>SNIR</b></p>

## Classe

Pour rappel, en langage objet, une propriété est une « variable d'objet », qui est définie dans une classe et dont la valeur est propre à chaque objet.

Une méthode est, en langage objet, une fonction définie dans une classe et qui sera propre à chaque objet.

En Java, créer une propriété se fait en indiquant la visibilité (private / protected / public) suivi du type et du nom de la propriété, comme une variable.

```
private String name;
```

En Java, créer une méthode se fait en indiquant la visibilité suivie du type de retour et du nom de la méthode. Si aucun retour n'est nécessaire, on utilisera le type virtuel « void ».

```
private int doubleIt(int bet) {
    return bet * 2;
}
```

Chaque argument doit être précédé de son type.

### 11. Créez une nouvelle classe « Character ». Cette classe doit définir les propriétés publiques name et age, respectivement une chaîne de caractères et un entier.

Copiez-collez le code ci-contre en bas de votre classe principale.

```
private static ArrayList<Character> characters = new ArrayList<Character>();
private static void addCharacter(Character c) {
    characters.add(c);
}
private static void printCharacters() {
    for(int i = 0; i < characters.size(); i++) {
        System.out.println(
            characters.get(i).name
            + " is "
            + String.valueOf(characters.get(i).age)
            + " years old"
        );
    }
}
```

*Attention, vous aurez besoin d'importer la classe correspondante pour ArrayList : utilisez votre IDE.*

Modifiez le code de votre classe principale, pour que celui-ci demande le nom de l'utilisateur. Puis, si le nom est non-vide, son âge. Si l'âge est supérieur à 0, instanciez un nouvel objet Character en utilisant new, et utilisez la méthode addCharacter(). Si le nom ou l'âge est vide, terminez l'exécution du programme en utilisant la méthode printCharacters().

```
Character c = new Character();
c.name = whatTheUserTypedIn;
addCharacter(c);
```

### 12. Effectuez les modifications demandées et testez votre code.

	<p><b>Java</b></p>	<p><b>BTS</b></p>
<p><b>TP</b></p>	<p><b>Blackjack</b></p>	<p><b>SNIR</b></p>

## Blackjack

### Début du jeu

Créez un nouveau projet « blackjack ».

Créez une classe Card. Chaque objet de cette classe représentera une carte unique, et possédera donc une valeur et une couleur.

Implémentez la classe Card comportant les propriétés suivantes :

- color : un caractère parmi les valeurs suivantes :
  - H (pour Cœur)
  - D (pour Carreau)
  - S (pour Pique)
  - C (pour Trèfle)
- value : un entier de 1 à 13 :
  - 1 pour l'As
  - 2-10 pour la valeur indiquée
  - 11 pour le Valet
  - 12 pour la Dame
  - 13 pour le Roi

### 13. Effectuez la création de la classe.

En Java, un constructeur est ajouté à une classe en ajoutant une méthode sans type de retour, du même nom que la classe. Un constructeur pour notre classe Character serait par exemple ci-contre.

Pour rappel, un constructeur est une méthode spéciale appelée à l'instanciation d'un objet.

Une autre méthode spéciale est toString. La méthode toString, si elle est définie, permet de renvoyer la représentation en chaîne de caractères de l'objet. Cela peut être utile pour afficher l'objet, pour le debug ou pour l'utilisateur.

```
public class Character {
    public String name;

    public Character() {
    }

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

Ajoutez un constructeur à votre classe. Ce constructeur prendra deux arguments color et value, tous deux entiers. Ce constructeur initialisera value et color, en traduisant l'entier passé en argument selon la règle suivante :

Argument	0	1	2	3
Couleur de carte	Coeur	Carreau	Pique	Trèfle

Ajoutez une méthode toString qui renverra une chaîne de caractères décrivant la carte, par exemple, pour une dame de pique, toString renverra « S12 ».

Enfin, créez votre classe principale « Blackjack » avec un main, et collez-y le code suivant :

```
Card c0 = new Card(0,1);
Card c1 = new Card(1,10);
Card c2 = new Card(2,11);
Card c3 = new Card(3,13);

System.out.println(c0);
System.out.println(c1);
System.out.println(c2);
System.out.println(c3);
```

#### 14. Testez et validez le code avec votre enseignant.

	<b>Java</b>	<b>BTS</b>
<b>TP</b>	<b>Blackjack</b>	<b>SNIR</b>

### Liste de cartes

Implémentez une classe Deck, qui contiendra les propriétés suivantes :

- content : un tableau de 52 cartes

La classe doit avoir un constructeur explicite ne prenant aucun argument mais devant remplir content avec toutes les cartes possibles.

Ajoutez une méthode toString pour afficher la liste complète des cartes, séparées par une virgule.

Enfin, ajoutez une méthode getCard() prenant un argument entier i, renvoyant la carte à la position i.

Modifiez votre classe principale afin d'y appliquer le code de test suivant :

```
Deck d = new Deck();
System.out.println(d);
System.out.println(d.getCard(15));
```

### 15. Réalisez le code et testez-le

### Swap de cartes

Ajoutez une méthode statique swap() qui ne retourne aucune valeur et prend deux arguments : deux entiers i et j.

Cette méthode devra échanger les cartes aux positions i et j dans la liste.

Modifiez votre classe principale afin d'y appliquer le code de test suivant :

```
Deck d = new Deck();
d.swap(8, 15);
d.swap(25, 7);
d.swap(15, 10);
System.out.println(d);
```

### 16. Réalisez le code et testez-le.

### Mélange

Ajouter, dans votre classe principale, la méthode getRandomNb() comme indiqué ci-dessous (vous devrez importer certaines classes) :

```
private static Random rnd = ThreadLocalRandom.current();
public static int getRandomNb(int max) {
    return rnd.nextInt(max);
}
```

Ajoutez une méthode statique shuffle() qui ne prend aucun argument, et doit mélanger aléatoirement les cartes.

Modifiez votre classe principale afin d'y appliquer le code de test suivant :

```
Deck d = new Deck();
d.shuffle();
System.out.println(d);
```

### 17. Réalisez le code et testez-le.

**Vérifiez votre progression auprès de votre enseignant.**

	<b>Java</b> <b>Blackjack</b>	<b>BTS</b>
		<b>SNIR</b>

## Evolution

Pour implémenter les fonctionnalités suivantes, nous allons avoir besoin de transformer notre tableau de Card, dans la classe Deck, en une collection. Une collection, en Java, est un objet permettant de gérer des listes, des tableaux et bien d'autres collections d'éléments.

Une classe gérant des collections est la classe ArrayList ; cette classe permet de gérer facilement une liste d'éléments. Le type d'objets contenus dans la liste est indiqué entre chevrons, comme dans l'exemple ci-dessous.

```
ArrayList<String> listOfStrings = new ArrayList<String>();
```

Vous aurez besoin de chercher de vous-même les informations et la documentation sur cette classe.

**18. Modifiez le code de la classe Deck et tout code nécessaire pour utiliser un ArrayList de Card en lieu et place d'un tableau de Card.**

## Pioche

**19. Dans votre classe Deck, implémentez une méthode « draw », sans argument, qui retournera la première carte de la pile, et la retirera de la liste contenue dans Deck. S'il n'y a plus de carte dans Deck, la méthode retournera null.**

## Classe Player

Implémentez une classe Player. Celle-ci contiendra les propriétés suivantes :

- Name (Chaîne de caractères)
- Money (Entier)
- Hand (ArrayList de Card)

Ainsi que les méthodes suivantes :

- int getValue() qui renvoie la valeur de la main possédée (pour le moment, nous renverrons toujours 0)
- void addCard(Card c) qui ajoute une carte à la Hand du Player
- void addMoney(int money) qui ajoute un montant
- void subMoney(int money) qui retire un montant

**20. Réalisez le code**

**Vérifiez votre avancée auprès de votre enseignant.**



	<b>Java</b>	<b>BTS</b>
<b>TP</b>	<b>Blackjack</b>	<b>SNIR</b>

## TP Autonome

### Contexte et règles

Les règles du Blackjack varient selon le type de casino, le pays... Nous allons fixer nous-mêmes certains points.

- Un seul joueur humain joue contre la banque (notre programme)
- L'objectif est d'obtenir un meilleur score que la banque sans dépasser 21
- Le joueur dépose une mise au début de la partie
- La banque distribue une carte au joueur et deux cartes à elle-même
- Toutes les cartes sont face visible
- La banque demande au joueur s'il veut une carte.
- Tant que le joueur veut une carte, la banque lui en tire une face visible.
- Si le joueur dépasse 21, le joueur perd immédiatement sa mise et la partie.
- Si le joueur ne dépasse pas 21 et décide d'arrêter de prendre des cartes, la banque tire des cartes jusqu'à dépasser 16.
- Si la banque dépasse 21, elle perd, et le joueur remporte l'équivalent de sa mise.
- Sinon, si la banque dépasse le joueur, le joueur perd sa mise.
- Sinon, si la banque équivaut le joueur, le joueur récupère sa mise.
- Sinon, si la banque est plus basse que le joueur, le joueur remporte l'équivalent de sa mise.

Les règles de valeur des cartes sont décidées comme suit :

- Chaque carte de 2 à 9 vaut la valeur inscrite
- Chaque figure (valet, dame, roi) et le 10 vaut 10
- L'as vaut 1 ou 11 en fonction de ce qui arrange la personne qui le possède

### Implémentation

**Implémentez le blackjack en vous servant du code que nous avons commencé à implémenter.**

Quelques précisions et aides :

- La valeur de la mise de départ doit être placée une seule fois dans le programme. Vous pouvez utiliser une constante ou une propriété statique de la classe principale.
- Créez autant de méthodes et de propriétés dont vous avez besoin.
- La banque est considérée comme une classe Player. Vous aurez donc deux instances de Player : la banque et le joueur humain.
- L'as vaut ce qui arrange le Player qui le possède. Si un ou plusieurs as sont dans la main du joueur, faites le calcul pour chaque valeur possible (1 ou 11) et prenez la valeur la plus haute qui ne dépasse pas 21.
- Pour laisser le joueur humain choisir s'il veut une nouvelle carte ou non, utilisez la ligne de commande en demandant au joueur de taper une lettre, par exemple.

## Annexes

### A1 - Diagramme Sysml State Machine

