PARTIEL C#1 : DAB

Consignes de rendu

Archive

Vous devez soumettre votre travail en suivant l'architecture suivante :
(en remplaçant "prenom.nom" par votre login)

```
42sh$ ls
prenom.nom    prenom.nom.zip
42sh$ tree prenom.nom
prenom.nom
|-- Bases
|   |-- Bases
|   |   |-- Bases.csproj
|   |   |-- Program.cs
|   |   |-- Properties
|   |   |   |-- AssemblyInfo.cs
|   |-- Bases.sln
|-- DAB
    |-- DAB
    |   |-- Bank.cs
    |   |-- DAB.cs
    |   |-- DAB.csproj
    |   |-- Inputs.cs
    |   |-- Menus.cs
    |   |-- Program.cs
    |   |-- Properties
    |   |   |-- AssemblyInfo.cs
    |   |-- User.cs
    |-- DAB.sln

6 directories, 13 files
```

— Votre code **DOIT** compiler.
— Votre architecture de fichier **DOIT** correspondre à celle ci-dessus.

Submission

Pour **soumettre votre travail**, vous devez créer un fichier zip (en remplaçant "prenom.nom" par votre login) et l'upload sur le site `https://exam-sup.pie.cri.epita.fr` :

```
Methode 1 (Terminal)
--------------------
42sh$ ls
prenom.nom
42sh$ zip -r prenom.nom.zip prenom.nom
42sh$ ls
prenom.nom    prenom.nom.zip


Methode 2 (Thunar)
--------------------
- Win (ou Alt) + d    (ouvrir le dmenu)
- Entrer "thunar" et appuyer sur la touche {Enter}
- Naviguer vers votre dossier "prenom.nom"
- Clic droit sur votre dossier "prenom.nom"
- Cliquer sur "Create archive"
- Selectionner "zip" en bas a gauche
- Valider en bas a droite
```

Vous pouvez **vérifier tous les fichiers soumis** grâce à l'affichage de l'architecture du contenu de votre zip, après l'avoir upload sur le site `https://exam-sup.pie.cri.epita.fr`.

Sujet

Le sujet est disponible sur le site `https://exam-sup.pie.cri.epita.fr`. Vous pouvez l'ouvrir à l'aide de la commande suivante :

```
42sh$ evince subject.pdf &
```

Faites attention

En lançant votre programme, si vous obtenez l'erreur **Deprecated : ISO-Latin1 characters in identifiers**, c'est surement que vous avez essayé de copier/coller des accents ou des caractères spéciaux qui ne sont pas valides !

1 Bases

1.1 HelloName

— **Fichier :** Bases/Bases/Program.cs

— **Description :** Affiche la chaîne de caractères "Hello name!" avec un nom personnalisé.

— **Paramètres :**
- **name :** Le nom à mettre dans la chaîne de caractères.

— **Fonctions autorisées :** Toutes

```
1  public static void HelloName(string name);
```

Exemples :

```
1  HelloName("");
2  // Hello World!
3  HelloName("ACDC");
4  // Hello ACDC!
```

1.2 PrintFibo

— **Fichier :** Bases/Bases/Program.cs

— **Description :** Affiche la suite de Fibonacci, une valeur par ligne.

— **Paramètres :**
- **n :** La position du dernier nombre de la suite, en commençant à zéro.

— **Fonctions autorisées :** Toutes

— **Interdit :** Vous n'êtes pas autorisés à utiliser la récurrence pour cet exercice.

```
1  public static void PrintFibo(uint n);
```

Exemples :

```
1  PrintFibo(0);
2  // 0
3
4  PrintFibo(6);
5  // 0
6  // 1
7  // 1    (= 0 + 1)          Formule:
8  // 2    (= 1 + 1)            fibo(0) = 0
9  // 3    (= 1 + 2)            fibo(1) = 1
10 // 5    (= 2 + 3)            fibo(n) = fibo(n - 1) + fibo(n - 2)
11 // 8    (= 3 + 5)
```

1.3 ItoaBase

— **Fichier :** Bases/Bases/Program.cs

— **Description :** Convertit un entier non signé en une chaîne de caractères dans la base spécifiée.

— **Paramètres :**
- **n :** Le nombre à convertir.
- **b :** La base dans laquelle le nombre doit être convertit.

— **Fonctions autorisées :** Toutes

```
enum BASE
{
    BINARY = 2,
    OCTAL = 8,
    DECIMAL = 10
};

public static string ItoaBase(uint n, BASE b);
```

Exemples :

```
Console.WriteLine(ItoaBase(42, BINARY));
// 101010
Console.WriteLine(ItoaBase(42, OCTAL));
// 52
Console.WriteLine(ItoaBase(42, DECIMAL));
// 42
```

ATTENTION

Vous n'êtes PAS autorisés à concaténer un *int* ou un *uint* à une *string*.
Vous DEVEZ seulement concaténer un *char* avec une *string*.

```
return 42 + "";          // Interdit
return '4' + '2' + "";   // Autorisé
```

Tip - Convertir une énumération en entier non signé

```
BASE b = DECIMAL;
uint num = (uint)b;
// num == 10
```

Tip - Convertir un entier non signé en un caractère

```
uint n = 4;
char c = (char)('0' + n);
// c == '4'
```

2 DAB

Pour cet exercice, vous allez implémenter un DAB "distributeur automatique de billet". Ne soyez pas effrayés par le nombre de pages ! Les fonctions que vous avez à implémenter sont courtes et très détaillées. Pour cet exercice, une tarball vous est donnée.

Un DAB ne gère qu'une banque. Une banque gère plusieurs utilisateurs. Un utilisateur peut ajouter/retirer de l'argent de sa banque uniquement en utilisant un DAB de sa banque en faisant respectivement un deposit(débôt)/withdraw(retrait).

2.1 Classes

2.1.1 User class

— **Fichier :** DAB/DAB/User.cs

— **Description :** Cette classe représente un utilisateur.

```
1  public class User
2  {
3      // Le nom de l'utilisateur
4      public string name { get; }
5      // La banque de l'utilisateur
6      public Bank bank { get; private set; }
7
8      public User(string name);
9      public void JoinBank(Bank bank);
10     public void DisplayUserInfo();
11 }
```

2.1.2 Bank class

— **Fichier :** DAB/DAB/Bank.cs

— **Description :** Cette classe représente une banque.

```
1  public class Bank
2  {
3      // Le nom de la banque
4      public string name { get; }
5      // Tous les utilisateurs et leur argent
6      // /!\ Un montant d'argent de 2042 correspond
7      //     à une valeur de 20.42EUR
8      private Dictionary<User, uint> users;
9
10     public Bank(string name);
11     public void AddUser(User user);
12     public uint GetMoneyOf(User user);
13     public bool AllowWithdraw(User user, uint amount);
14     public void Deposit(User user, uint amount);
15     public void Withdraw(User user, uint amount);
16     public void DisplayBankInfo();
17 }
```

2.1.3 DAB class

— **Fichier :** DAB/DAB/Bank.cs

— **Description :** Cette classe représente un DAB.

```
1  public class DAB
2  {
3      // L'identifiant unique du DAB
4      public uint uid { get; }
5      // Nombre de DAB en circulation
6      private static uint dabCount;
7
8      // La banque associée au DAB
9      public readonly Bank bank;
10     // L'argent qu'il reste dans le DAB
11     // /!\ Un montant d'argent de 2042 correspond
12     //     à une valeur de 20.42EUR
13     private uint moneyLeft { get; set; }
14
15     public DAB(Bank bank);
16     public void Fill(uint amount);
17     public bool AllowWithdraw(User user, uint amount);
18     public void Withdraw(User user, uint amount);
19     public bool AllowDeposit(User user);
20     public void Deposit(User user, uint amount);
21     public void DisplayDABInfo();
22 }
```

2.2 Constructeurs

2.2.1 User

— **Fichier :** DAB/DAB/User.cs

— **Description :** Initialise un nouvel utilisateur.

— **Paramètres :**
- **name :** Le nom de l'utilisateur.

— **Fonctions autorisées :** Toutes

```
1  public User(string name);
```

Exemples :

```
1  User user = new User("Junior");
2  Console.WriteLine("user.name is equal to: " + user.name);
3  // user.name is equal to: Junior
```

2.2.2 Bank

— **Fichier :** DAB/DAB/Bank.cs

— **Description :** Initialise une nouvelle banque.

— **Paramètres :**
- **name :** Le nom de la banque.

— **Fonctions autorisées :** Toutes

```
1  public Bank(string name);
```

Exemples :

```
1  Bank bank = new Bank("EPITA");
2  Console.WriteLine("bank.name is equal to: " + bank.name);
3  // bank.name is equal to: EPITA
```

2.2.3 DAB

— **Fichier :** DAB/DAB/DAB.cs

— **Description :** Initialise un nouveau DAB.

— **Paramètres :**
- **bank :** La banque associée au DAB.

— **Fonctions autorisées :** Toutes

```
1  public DAB(Bank bank);
```

Exemples :

```
1  Console.WriteLine("DAB.dabCount is equal to: " + DAB.dabCount);
2  // DAB.dabCount is equal to: 0
3
4  Bank bank = new Bank("EPITA");
5  DAB dab = new DAB(bank);
6
7  Console.WriteLine("dab.bank.name is equal to: " + dab.bank.name);
8  // dab.bank.name is equal to EPITA
9
10 Console.WriteLine("dab.bank.uid is equal to: " + dab.bank.uid);
11 // dab.bank.uid is equal to: 0
12
13 Console.WriteLine("DAB.dabCount is equal to: " + DAB.dabCount);
14 // DAB.dabCount is equal to: 1
```

2.3 Inputs

2.3.1 GetString

— **Fichier :** DAB/DAB/Inputs.cs

— **Description :** Tant que l'entrée est vide, écrire "texte : " avec un texte personnalisé et lire l'entrée.

— **Paramètres :**
- **text :** Le texte à afficher.

— **Fonctions autorisées :** Toutes

```
1 public static string GetString(string text);
```

Exemples :

```
1 string str = GetString("Test");
2 // Test: {Enter key press}
3 // Test: Something{Enter key press}
4
5 Console.WriteLine("str is equal to: " + str);
6 // str is equal to: Something
```

2.3.2 GetUInt

— **Fichier :** DAB/DAB/Inputs.cs

— **Description :** Tant que l'entrée n'est pas un entier non signé, écrire "texte : " avec un texte personnalisé et lire l'entrée.

— **Paramètres :**
- **text :** Le texte à afficher.

— **Fonctions autorisées :** Toutes

```
1 public static uint GetUInt(string text);
```

Exemples :

```
1 uint num = GetUInt("Test");
2 // Test: {Enter key press}
3 // Test: Something{Enter key press}
4 // Test: -1{Enter key press}
5 // Test: 42{Enter key press}
6
7 Console.WriteLine("num is equal to: " + num);
8 // num is equal to: 42
```

2.3.3 GetUser

— **Fichier :** DAB/DAB/Inputs.cs

— **Description :** Lit l'entrée tant que la ligne donnée ne correspond pas au nom d'un utilisateur.

— **Fonctions autorisées :** Toutes

```
1 public static User GetUser();
```

Exemples :

```
1 User user = GetUser();
2 // User name: {Enter key press}
3 // User name: ACDC{Enter key press}
4 // User name: Sophie{Enter key press}
5
6 Console.WriteLine("user.name is equal to: " + user.name);
7 // user.name is equal to: Sophie
```

2.3.4 GetDAB

— **Fichier :** DAB/DAB/Inputs.cs

— **Description :** Lit l'entrée tant que la ligne donnée ne correspond pas à l'uid d'un DAB.

— **Fonctions autorisées :** Toutes

```
1 public static DAB GetDAB();
```

Exemples :

```
1 DAB dab = GetDAB();
2 // DAB uid: {Enter key press}
3 // DAB uid: Something{Enter key press}
4 // DAB uid: -1{Enter key press}
5 // DAB uid: 0{Enter key press}
6
7 Console.WriteLine("dab.uid is equal to: " + dab.uid);
8 // dab.uid is equal to: 0
```

2.3.5 GetBank

— **Fichier :** DAB/DAB/Inputs.cs

— **Description :** Lit l'entrée tant que la ligne donnée ne correspond pas au nom d'une banque.

— **Fonctions autorisées :** Toutes

```
1 public static Bank GetBank();
```

Exemples :

```
1 Bank bank = GetBank();
2 // Bank name: {Enter key press}
3 // Bank name: ASM{Enter key press}
4 // Bank name: ACDC{Enter key press}
5
6 Console.WriteLine("bank.name is equal to: " + bank.name);
7 // bank.name is equal to: ACDC
```

2.4 User

2.4.1 JoinBank

— **Fichier :** DAB/DAB/User.cs

— **Description :** Définit la banque d'un utilisateur.

— **Paramètres :**
- **bank :** La banque pour l'utilisateur.

— **Fonctions autorisées :** Toutes

```
1 public void JoinBank(Bank bank);
```

Exemples :

```
1 User user = Inputs.GetUser();
2 // User name: Sophie{Enter key press}
3 Bank bank = Inputs.GetBank();
4 // Bank name: EPITA{Enter key press}
5
6 user.JoinBank(bank);
7 Console.WriteLine("user.bank is equal to: " + user.bank);
8 // user.bank is equal to EPITA
```

2.4.2 DisplayUserInfo

— **Fichier :** DAB/DAB/User.cs

— **Description :** Affiche les informations de l'utilisateur.

— **Fonctions autorisées :** Toutes

```
1  public void DisplayUserInfo();
```

Exemples :

```
1  User user = Inputs.GetUser();
2  // User name: Sophie{Enter key press}
3  user.DisplayUserInfo();
4
5  // === User Information ===
6  // Name: Sophie
7  // Bank: EPITA
8  // Money: 0.0EUR
```

2.5 Bank

2.5.1 GetMoneyOf

— **Fichier :** DAB/DAB/Bank.cs

— **Description :** Obtenir la quantité d'argent d'un utilisateur.

— **Paramètres :**
- **user :** L'utilisateur dont on veut la quantité d'argent.

— **Fonctions autorisées :** Toutes

```
1  public uint GetMoneyOf(User user);
```

Exemples :

```
1  User user = Inputs.GetUser();
2  // User name: Sophie{Enter key press}
3  Bank bank = Inputs.GetBank();
4  // Bank name: EPITA{Enter key press}
5  Console.WriteLine("Sophie's money: " + bank.getMoneyOf(user));
6  // Sophie's money: 0
```

2.5.2 AddUser

— **Fichier :** DAB/DAB/Bank.cs

— **Description :** Ajouter un nouvel utilisateur dans le dictionnaire et faire rejoindre la banque à l'utilisateur.

— **Paramètres :**
- **user :** L'utilisateur ajouté.

— **Fonctions autorisées :** Toutes

```
public void AddUser(User user);
```

Exemples :

```
User user = Inputs.GetUser();
// User name: Sophie{Enter key press}
Bank bank = Inputs.GetBank("EPITA");
// Bank name: EPITA{Enter key press}

bank.AddUser(user);

Console.WriteLine("bank[user] is equal to: " + bank[user]);
// bank[user] is equal to: 0
Console.WriteLine("user.bank is equal to: " + user.bank);
// user.bank is equal to: EPITA
```

2.5.3 AllowWithdraw

— **Fichier :** DAB/DAB/Bank.cs

— **Description :** Vérifier que l'utilisateur a assez d'argent dans la banque pour retirer le montant demandé par l'utilisateur.

— **Paramètres :**
- **user :** L'utilisateur qui veut retirer de l'argent.
- **amount** Le montant que l'utilisateur veut retirer.

— **Fonctions autorisées :** Toutes

```
public bool AllowWithdraw(User user, uint amount);
```

Exemples :

```
User user = Inputs.GetUser();
// User name: Sophie{Enter key press}
Bank bank = Inputs.GetBank();
// Bank name: EPITA{Enter key press}

Console.WriteLine("Sophie's money: " + bank.GetMoneyOf(user));
// Sophie's money: 0

Console.WriteLine("allow withdraw ? " + bank.AllowWithdraw(user, 2000));
// allow withdraw ? False
```

2.5.4 Deposit

— **Fichier :** DAB/DAB/Bank.cs

— **Description :** Dépose de l'argent dans le compte bancaire d'un utilisateur.

— **Paramètres :**
- **user :** L'utilisateur qui veut déposer de l'argent.
- **amount** Le montant que l'utilisateur veut déposer.

— **Fonctions autorisées :** Toutes

```
1  public void Deposit(User user, uint amount);
```

Exemples :

```
1  User user = Inputs.GetUser();
2  // User name: Sophie{Enter key press}
3  Bank bank = Inputs.GetBank();
4  // Bank name: EPITA{Enter key press}
5
6  Console.WriteLine("Sophie's money: " + bank.GetMoneyOf(user));
7  // Sophie's money: 0
8
9  bank.Deposit(user, 1000);
10
11 Console.WriteLine("Sophie's money: " + bank.GetMoneyOf(user));
12 // Sophie's money: 1000
```

2.5.5 Withdraw

— **Fichier :** DAB/DAB/Bank.cs

— **Description :** Retirer un montant du compte en banque d'un utilisateur.

— **Paramètres :**
- **user :** L'utilisateur qui veut retirer de l'argent.
- **amount** Le montant que l'utilisateur veut retirer.

— **Fonctions autorisées :** Toutes

```
public void Withdraw(User user, uint amount);
```

Exemples :

```
User user = Inputs.GetUser();
// User name: Sophie{Enter key press}
Bank bank = Inputs.GetBank();
// Bank name: EPITA{Enter key press}
bank.Deposit(user, 1000);

Console.WriteLine("Sophie's money: " + bank.GetMoneyOf(user));
// Sophie's money: 1000

bank.Withdraw(user, 100);

Console.WriteLine("Sophie's money: " + bank.GetMoneyOf(user));
// Sophie's money: 900
```

2.5.6 DisplayBankInfo

— **Fichier :** DAB/DAB/Bank.cs

— **Description :** Affiche les informations de la banque.

— **Fonctions autorisées :** Toutes

```
public void DisplayBankInfo();
```

Exemples :

```
Bank bank = Inputs.GetBank();
// Bank name: EPITA{Enter key press}

user.DisplayUserInfo();
// === Bank Information ===
// Name: EPITA
// Users:
// - Sophie: 0.0EUR
```

2.6 DAB

2.6.1 Fill

— **Fichier :** DAB/DAB/DAB.cs

— **Description :** Remplir le DAB avec de l'argent.

— **Paramètres :**
- **amount :** Le montant.

— **Fonctions autorisées :** Toutes

```
public void Fill(uint amount);
```

Exemples :

```
DAB dab = Inputs.GetDAB();
// DAB uid: 0{Enter key press}

Console.WriteLine("Money left in the DAB: " + dab.moneyLeft);
// Money left in the DAB: 30000

dab.Fill(1000);

Console.WriteLine("Money left in the DAB: " + dab.moneyLeft);
// Money left in the DAB: 31000
```

2.6.2 AllowWithdraw

— **Fichier :** DAB/DAB/DAB.cs

— **Description :** Vérifie que la banque du DAB correspond à celle de l'utilisateur, que le DAB contient assez d'argent et que la banque autorise l'utilisateur à retirer le montant.

— **Paramètres :**
- **user :** L'utilisateur qui veut retirer de l'argent.
- **amount :** Le montant que l'utilisateur veut retirer.

— **Fonctions autorisées :** Toutes

```
public bool AllowWithdraw(User user, uint amount);
```

Exemples :

```
User user = Inputs.GetUser();
// User name: Sophie{Enter key press}
DAB dab = Inputs.GetDAB();
// DAB uid: 0{Enter key press}

Console.WriteLine("dab.bank.name is equal to: " + dab.bank.name);
// dab.bank.name is equal to: EPITA
Console.WriteLine("Sophie's money: " + dab.bank.GetMoneyOf(user));
// Sophie's money: 0

Console.WriteLine("allow withdraw ? " + dab.AllowWithdraw(user, 2000));
// allow withdraw ? False
```

2.6.3 Withdraw

— **Fichier :** DAB/DAB/DAB.cs

— **Description :** Retire un montant du compte en banque d'un utilisateur et du DAB.

— **Paramètres :**
- **user :** L'utilisateur qui veut retirer de l'argent.
- **amount :** Le montant que l'utilisateur veut retirer.

— **Fonctions autorisées :** Toutes

```
1  public void Withdraw(User user, uint amount);
```

Exemples :

```
1  User user = Inputs.GetUser();
2  // User name: Sophie{Enter key press}
3  DAB dab = Inputs.GetDAB();
4  // DAB uid: 0{Enter key press}
5  dab.bank.Deposit(user, 1000);
6
7  Console.WriteLine("Sophie's money: " + dab.bank.GetMoneyOf(user));
8  // Sophie's money: 1000
9  Console.WriteLine("Money left in the DAB: " + dab.moneyLeft);
10 // Money left in the DAB: 30000
11
12 dab.Withdraw(user, 100);
13
14 Console.WriteLine("Sophie's money: " + dab.bank.GetMoneyOf(user));
15 // Sophie's money: 900
16 Console.WriteLine("Money left in the DAB: " + dab.moneyLeft);
17 // Money left in the DAB: 29900
```

2.6.4 AllowDeposit

— **Fichier :** DAB/DAB/DAB.cs

— **Description :** Vérifie que la banque du DAB correspond à celle de l'utilisateur qui veut déposer de l'argent.

— **Paramètres :**
- **user :** L'utilisateur qui veut déposer de l'argent.

— **Fonctions autorisées :** Toutes

```
public bool AllowDeposit(User user);
```

Exemples :

```
User user = Inputs.GetUser();
// User name: Sophie{Enter key press}
DAB dab = Inputs.GetDAB();
// DAB uid: 0{Enter key press}

Console.WriteLine("dab.bank.name is equal to: " + dab.bank.name);
// dab.bank.name is equal to: EPITA
Console.WriteLine("user.bank.name is equal to: " + user.bank.name);
// user.bank.name is equal to: EPITA

Console.WriteLine("allow deposit ? " + bank.AllowDeposit(user));
// allow deposit ? True
```

2.6.5 Deposit

— **Fichier :** DAB/DAB/DAB.cs

— **Description :** Dépose un montant sur le compte de l'utilisateur et dans le DAB.

— **Paramètres :**
- **user :** L'utilisateur qui veut déposer de l'argent.
- **amount :** Le montant que l'utilisateur veut déposer.

— **Fonctions autorisées :** Toutes

```
public void Deposit(User user, uint amount);
```

Exemples :

```
User user = Inputs.GetUser();
// User name: Sophie{Enter key press}
DAB dab = Inputs.GetDAB();
// DAB uid: 0{Enter key press}

Console.WriteLine("Sophie's money: " + dab.bank.GetMoneyOf(user));
// Sophie's money: 0
Console.WriteLine("Money left in the DAB: " + dab.moneyLeft);
// Money left in the DAB: 30000

dab.Deposit(user, 1000);

Console.WriteLine("Sophie's money: " + dab.bank.GetMoneyOf(user));
// Sophie's money: 1000
Console.WriteLine("Money left in the DAB: " + dab.moneyLeft);
// Money left in the DAB: 31000
```

2.6.6 DisplayDABInfo

— **Fichier :** DAB/DAB/DAB.cs

— **Description :** Affiche les informations du DAB.

— **Fonctions autorisées :** Toutes

```
public void DisplayDABInfo();
```

Exemples :

```
DAB dab = Inputs.GetDAB();
// DAB uid: 0{Enter key press}

dab.DisplayDABInfo();
// === DAB Information ===
// UID: 0
// Bank: EPITA
// Money left: 300.0EUR
```

2.7 Handling

2.7.1 HandleDeposit

— **Fichier :** DAB/DAB/Program.cs

— **Description :** Demande un utilisateur, un DAB et un montant et essaye de déposer le montant demandé.

— **Fonctions autorisées :** Toutes

```
1  private static void HandleDeposit();
```

Exemples :

```
1  Inputs.Init();
2  HandleDeposit();
3  // User name: Sophie{Enter key press}
4  // DAB uid: 0{Enter key press}
5  // Amount: 100{Enter key press}
6  // Operation successful!
7
8  HandleDeposit();
9  // User name: Sophie{Enter key press}
10 // DAB uid: 2{Enter key press}
11 // Amount: 100{Enter key press}
12 // Error: Wrong bank.
```

2.7.2 HandleWithdraw

— **Fichier :** DAB/DAB/Program.cs

— **Description :** Demande un utilisateur, un DAB et un montant et essaye de retirer le montant demandé.

— **Fonctions autorisées :** Toutes

```
1  private static void HandleWithdraw();
```

Exemples :

```
1  Inputs.Init();
2  User user = Inputs.GetUser();
3  // User name: Sophie{Enter key press}
4  DAB dab = Inputs.GetDAB();
5  // DAB uid: 0{Enter key press}
6  dab.Deposit(user, 100);
7
8  HandleWithdraw();
9  // User name: Sophie{Enter key press}
10 // DAB uid: 0{Enter key press}
11 // Amount: 100{Enter key press}
12 // Operation successful!
13
14 HandleWithdraw();
15 // User name: Sophie{Enter key press}
16 // DAB uid: 0{Enter key press}
17 // Amount: 100{Enter key press}
18 // Error: Wrong bank or not enough money
19
20 HandleWithdraw();
21 // User name: Sophie{Enter key press}
22 // DAB uid: 2{Enter key press}
23 // Amount: 100{Enter key press}
24 // Error: Wrong bank or not enough money
```

2.7.3 HandleUserInfo

— **Fichier :** DAB/DAB/Program.cs

— **Description :** Demande le nom d'un utilisateur et affiche ses informations.

— **Fonctions autorisées :** Toutes

```
1 private static void HandleUserInfo();
```

Exemples :

```
1 Inputs.Init();
2
3 HandleUserInfo();
4 // User name: Jojo{Enter key press}
5 // User name: Sophie{Enter key press}
6 // === User Information ===
7 // Name: Sophie
8 // Bank: EPITA
9 // Money: 0
```

2.7.4 HandleDABInfo

— **Fichier :** DAB/DAB/Program.cs

— **Description :** Demande l'uid d'un DAB et affiche ses informations.

— **Fonctions autorisées :** Toutes

```
1 private static void HandleDABInfo();
```

Exemples :

```
1 Inputs.Init();
2
3 HandleDABInfo();
4 // DAB uid: 2000{Enter key press}
5 // DAB uid: 0{Enter key press}
6 // === DAB Information ===
7 // UID: 0
8 // Bank: EPITA
9 // Money left: 300.0EUR
```

2.7.5 HandleBankInfo

— **Fichier :** DAB/DAB/Program.cs

— **Description :** Demande le nom d'une banque et affiche ses informations.

— **Fonctions autorisées :** Toutes

```
1  private static void HandleBankInfo();
```

Exemples :

```
1  Inputs.Init();
2
3  HandleBankInfo();
4  // Bank name: ASM{Enter key press}
5  // Bank name: ACDC{Enter key press}
6  // === Bank Information ===
7  // Name: ACDC
8  // Users:
9  // - Calcifer: 0.0EUR
10 // - Navet: 0.0EUR
```

I see no point in coding if it can't be beautiful.