# PARTIEL C#1: DAB

## Assignment

### Archive

You must submit your work with the following architecture :
(replacing "prenom.nom" with your 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
```

- Your code **MUST** compile.
- Your file architecture **MUST** match the one described above.

## Submission

To **submit your work**, you must create a zip file (replacing "prenom.nom" with your login) and upload it on the website `https://exam-sup.pie.cri.epita.fr` :

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


Method 2 (Thunar)
--------------------
- Win (ou Alt) + d     (open the dmenu)
- Type "thunar" and press on the {Enter} key
- Go to your folder "prenom.nom"
- Right click on your folder "prenom.nom"
- Click on "Create archive"
- Select "zip" in the botton left corner
- Validate in the bottom right corner
```

You can **check all your submitted files** using the display of the architecture of the content of your zip file, once it is uploaded on the website `https://exam-sup.pie.cri.epita.fr`.

## Subject

The subject is available on the website `https://exam-sup.pie.cri.epita.fr`. You can open it using the following command:

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

**Be careful**

When you run your program, if you get the error **Deprecated: ISO-Latin1 characters in identifiers**, it is surely because you tryed to copy/paste some accents or special characters that are not valid!

# 1 Bases

## 1.1 HelloName

- **File:** Bases/Bases/Program.cs
- **Description:** Prints the string "Hello name!" with a custom name.
- **Parameters:**
  - **name:** The name to place in the string.
- **Allowed functions:** All

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

Examples:

```
HelloName("");
// Hello World!
HelloName("ACDC");
// Hello ACDC!
```

## 1.2 PrintFibo

- **File:** Bases/Bases/Program.cs
- **Description:** Prints the Fibonacci sequence, one value per line.
- **Parameters:**
  - **n:** The position of the last number of the sequence to print, starting from zero.
- **Allowed functions:** All
- **Forbidden:** You are not allowed to use recursion for this exercise.

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

Examples:

```
PrintFibo(0);
// 0

PrintFibo(6);
// 0
// 1
// 1    (= 0 + 1)          Formula:
// 2    (= 1 + 1)            fibo(0) = 0
// 3    (= 1 + 2)            fibo(1) = 1
// 5    (= 2 + 3)            fibo(n) = fibo(n - 1) + fibo(n - 2)
// 8    (= 3 + 5)
```

### 1.3 ItoaBase

- **File:** Bases/Bases/Program.cs
- **Description:** Converts an unsigned integer to a string in the base specified.
- **Parameters:**
  - **n:** The number to convert.
  - **b:** The base the number must be converted in.
- **Allowed functions:** All

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

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

Examples:

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

**WARNING**

You are NOT allowed to append an *int* or *uint* with a *string*.
You MUST only append a *char* to a *string*.

```
return 42 + "";            // Forbidden
return '4' + '2' + "";     // Allowed
```

**Tip - Convert an enum into a uint**

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

**Tip - Convert a uint into a char**

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

# 2 DAB

For this exercise, you will implement a DAB "distributeur automatique de billet" (an ATM). Don't let the number of pages scare you! Every function you have to implement is short and is explained in great detail. For this exercice, you are given a tarball.

A DAB handles only one bank. A bank handles many users. A user can add/remove money to his bank by only using a DAB of his/her bank by respectively making a deposit/withdraw.

## 2.1 Classes

### 2.1.1 User class

- **File:** DAB/DAB/User.cs
- **Description:** This class represents a User.

```
1  public class User
2  {
3      // The name of the user
4      public string name { get; }
5      // The bank of the user
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

- **File:** DAB/DAB/Bank.cs
- **Description:** This class represents a Bank.

```
1  public class Bank
2  {
3      // The name of the bank
4      public string name { get; }
5      // All the users and their money
6      // /!\ A money amount equal to 2042 corresponds
7      //     to a value of 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

- **File:** DAB/DAB/Bank.cs
- **Description:** This class represents a DAB (an ATM).

```
public class DAB
{
    // The unique identifier of the DAB
    public uint uid { get; }
    // Store the total dab count
    private static uint dabCount;

    // The bank associated with the DAB
    public readonly Bank bank;
    // The money left in the DAB
    // /!\ A money amount equal to 2042 corresponds
    //     to a value of 20.42EUR
    private uint moneyLeft { get; set; }

    public DAB(Bank bank);
    public void Fill(uint amount);
    public bool AllowWithdraw(User user, uint amount);
    public void Withdraw(User user, uint amount);
    public bool AllowDeposit(User user);
    public void Deposit(User user, uint amount);
    public void DisplayDABInfo();
}
```

## 2.2 Constructors

### 2.2.1 User

- **File:** DAB/DAB/User.cs
- **Description:** Initialize a new User.
- **Parameters:**

  - **name:** The name of the user.

- **Allowed functions:** All

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

Examples:

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

### 2.2.2 Bank

- **File:** DAB/DAB/Bank.cs
- **Description:** Initialize a new Bank.
- **Parameters:**
  - **name:** The name of the bank.
- **Allowed functions:** All

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

Examples:

```
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

- **File:** DAB/DAB/DAB.cs
- **Description:** Initialize a new DAB.
- **Parameters:**
  - **bank:** The bank associated with the DAB.
- **Allowed functions:** All

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

Examples:

```
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

- **File:** DAB/DAB/Inputs.cs
- **Description:** While the input is empty, write "text: " with a custom text and read the input line.
- **Parameters:**

  - **text:** The text to display.
- **Allowed functions:** All

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

Examples:

```
string str = GetString("Test");
// Test: {Enter key press}
// Test: Something{Enter key press}

Console.WriteLine("str is equal to: " + str);
// str is equal to: Something
```

### 2.3.2 GetUInt

- **File:** DAB/DAB/Inputs.cs
- **Description:** While the input is not an unsigned integer, write "text: " with a custom text and read the input line.
- **Parameters:**

  - **text:** The text to display.
- **Allowed functions:** All

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

Examples:

```
uint num = GetUInt("Test");
// Test: {Enter key press}
// Test: Something{Enter key press}
// Test: -1{Enter key press}
// Test: 42{Enter key press}

Console.WriteLine("num is equal to: " + num);
// num is equal to: 42
```

### 2.3.3 GetUser

- **File:** DAB/DAB/Inputs.cs
- **Description:** Read the input line while the given string does not correspond to any user.
- **Allowed functions:** All

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

Examples:

```
User user = GetUser();
// User name: {Enter key press}
// User name: ACDC{Enter key press}
// User name: Sophie{Enter key press}

Console.WriteLine("user.name is equal to: " + user.name);
// user.name is equal to: Sophie
```

### 2.3.4 GetDAB

- **File:** DAB/DAB/Inputs.cs
- **Description:** Read the input line while the given number does not correspond to any DAB's uid.
- **Allowed functions:** All

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

Examples:

```
DAB dab = GetDAB();
// DAB uid: {Enter key press}
// DAB uid: Something{Enter key press}
// DAB uid: -1{Enter key press}
// DAB uid: 0{Enter key press}

Console.WriteLine("dab.uid is equal to: " + dab.uid);
// dab.uid is equal to: 0
```

### 2.3.5 GetBank

- **File:** DAB/DAB/Inputs.cs
- **Description:** Read the input line while the given string does not correspond to any bank.
- **Allowed functions:** All

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

Examples:

```
Bank bank = GetBank();
// Bank name: {Enter key press}
// Bank name: ASM{Enter key press}
// Bank name: ACDC{Enter key press}

Console.WriteLine("bank.name is equal to: " + bank.name);
// bank.name is equal to: ACDC
```

## 2.4 User

### 2.4.1 JoinBank

- **File:** DAB/DAB/User.cs
- **Description:** Set the bank of the user.
- **Parameters:**
  - **bank:** The bank for the user.
- **Allowed functions:** All

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

Examples:

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

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

### 2.4.2 DisplayUserInfo

- **File:** DAB/DAB/User.cs
- **Description:** Display information about the user.
- **Allowed functions:** All

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

Examples:

```
User user = Inputs.GetUser();
// User name: Sophie{Enter key press}
user.DisplayUserInfo();

// === User Information ===
// Name: Sophie
// Bank: EPITA
// Money: 0.0EUR
```

## 2.5 Bank

### 2.5.1 GetMoneyOf

- **File:** DAB/DAB/Bank.cs
- **Description:** Get the money amount of a user.
- **Parameters:**
  - **user:** The user whose amount of money we want to know.
- **Allowed functions:** All

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

Examples:

```
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
```

### 2.5.2 AddUser

- **File:** DAB/DAB/Bank.cs
- **Description:** Add a new user to the dictionary and have the user join the bank.
- **Parameters:**
  - **user:** The user added.
- **Allowed functions:** All

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

Examples:

```
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

- **File:** DAB/DAB/Bank.cs
- **Description:** Verify that the user has enough money in the bank to withdraw the amount asked by the user.
- **Parameters:**
  - **user:** The user wanting to withdraw money.
  - **amount** The amount of money the user wants to withdraw.
- **Allowed functions:** All

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

Examples:

```
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

- **File:** DAB/DAB/Bank.cs
- **Description:** Deposit an amount of money to the user bank account.
- **Parameters:**
  - **user:** The user wanting to deposit money.
  - **amount** The amount of money the user wants to deposit.
- **Allowed functions:** All

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

Examples:

```
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

bank.Deposit(user, 1000);

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

### 2.5.5 Withdraw

- **File:** DAB/DAB/Bank.cs
- **Description:** Withdraw an amount of money from the user bank account.
- **Parameters:**
  - **user:** The user waiting to withdraw money.
  - **amount** The amount of money the user wants to withdraw.
- **Allowed functions:** All

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

Examples:

```
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

- **File:** DAB/DAB/Bank.cs
- **Description:** Display information about the bank.
- **Allowed functions:** All

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

Examples:

```
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

- **File:** DAB/DAB/DAB.cs
- **Description:** Fill the DAB with money.
- **Parameters:**
  - **amount:** The amount of money.
- **Allowed functions:** All

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

Examples:

```
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

- **File:** DAB/DAB/DAB.cs
- **Description:** Verify that the DAB's bank is the same as the user's, that the DAB has enough money and that the bank allows the user to withdraw money from it.
- **Parameters:**
  - **user:** The user wanting to withdraw money.
  - **amount:** The amount of money the user wants to withdraw.
- **Allowed functions:** All

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

Examples:

```
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

- **File:** DAB/DAB/DAB.cs
- **Description:** Withdraw an amount of money from the user bank account and from the DAB.
- **Parameters:**
  - **user:** The user wanting to withdraw money.
  - **amount:** The amount of money the user wants to withdraw.
- **Allowed functions:** All

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

Examples:

```
User user = Inputs.GetUser();
// User name: Sophie{Enter key press}
DAB dab = Inputs.GetDAB();
// DAB uid: 0{Enter key press}
dab.bank.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: 30000

dab.Withdraw(user, 100);

Console.WriteLine("Sophie's money: " + dab.bank.GetMoneyOf(user));
// Sophie's money: 900
Console.WriteLine("Money left in the DAB: " + dab.moneyLeft);
// Money left in the DAB: 29900
```

### 2.6.4 AllowDeposit

- **File:** DAB/DAB/DAB.cs
- **Description:** Verify that the DAB's bank corresponds to the users's bank.
- **Parameters:**
  - **user:** The user wanting to deposit money.
- **Allowed functions:** All

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

Examples:

```
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

- **File:** DAB/DAB/DAB.cs
- **Description:** Deposit an amount of money to the user bank account and to the DAB.
- **Parameters:**
  - **user:** The user wanting to withdraw money.
  - **amount:** The amount of money the user wants to withdraw.
- **Allowed functions:** All

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

Examples:

```
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

- **File:** DAB/DAB/DAB.cs
- **Description:** Display information about the DAB.
- **Allowed functions:** All

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

Examples:

```
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

- **File:** DAB/DAB/Program.cs
- **Description:** Gets a user, a DAB and an amount and tries to deposit money into it.
- **Allowed functions:** All

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

Examples:

```
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

- **File:** DAB/DAB/Program.cs
- **Description:** Gets a user, a DAB and an amount and tries to withdraw money from it.
- **Allowed functions:** All

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

Examples:

```
Inputs.Init();
User user = Inputs.GetUser();
// User name: Sophie{Enter key press}
DAB dab = Inputs.GetDAB();
// DAB uid: 0{Enter key press}
dab.Deposit(user, 100);

HandleWithdraw();
// User name: Sophie{Enter key press}
// DAB uid: 0{Enter key press}
// Amount: 100{Enter key press}
// Operation successful!

HandleWithdraw();
// User name: Sophie{Enter key press}
// DAB uid: 0{Enter key press}
// Amount: 100{Enter key press}
// Error: Wrong bank or not enough money

HandleWithdraw();
// User name: Sophie{Enter key press}
// DAB uid: 2{Enter key press}
// Amount: 100{Enter key press}
// Error: Wrong bank or not enough money
```

### 2.7.3 HandleUserInfo

- **File:** DAB/DAB/Program.cs
- **Description:** Gets a user and displays his information.
- **Allowed functions:** All

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

Examples:

```
Inputs.Init();

HandleUserInfo();
// User name: Jojo{Enter key press}
// User name: Sophie{Enter key press}
// === User Information ===
// Name: Sophie
// Bank: EPITA
// Money: 0
```

### 2.7.4 HandleDABInfo

- **File:** DAB/DAB/Program.cs
- **Description:** Gets a DAB and displays his information.
- **Allowed functions:** All

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

Examples:

```
Inputs.Init();

HandleDABInfo();
// DAB uid: 2000{Enter key press}
// DAB uid: 0{Enter key press}
// === DAB Information ===
// UID: 0
// Bank: EPITA
// Money left: 300.0EUR
```

### 2.7.5 HandleBankInfo

- **File:** DAB/DAB/Program.cs
- **Description:** Gets a bank and displays his information.
- **Allowed functions:** All

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

Examples:

```
Inputs.Init();

HandleBankInfo();
// Bank name: ASM{Enter key press}
// Bank name: ACDC{Enter key press}
// === Bank Information ===
// Name: ACDC
// Users:
// - Calcifer: 0.0EUR
// - Navet: 0.0EUR
```

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