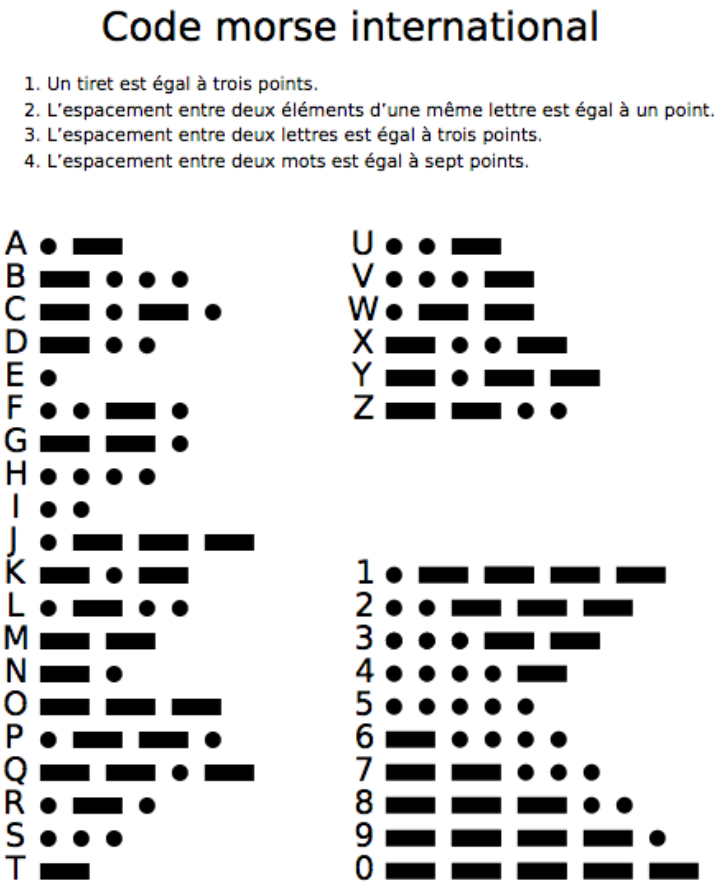
PROJET N°3 ET 4 *Le code Morse*

Objectifs

Coder en Java un convertisseur de chaîne de caractère vers morse qui fonctionne pour des textes écrits en majuscule, sans accents ni chiffres.

« *L'alphabet morse ou code morse, est un code permettant de transmettre un texte à l'aide de séries d'impulsions courtes et longues, qu'elles soient produites par des signes, une lumière, un son ou un geste.* » (Wikipedia.)



Exercice 1 *Modélisation d'un signal en morse*

On décide de modéliser une transmission en morse par une chaîne de caractère :

- une impulsion courte (un point) sera représentée par le symbole `=`
- une impulsion longue (un tiret) sera représentée par le symbole `===`
- l'absence de signal (espacement entre deux signaux d'une même lettre) sera représentée par le symbole `_`

Par exemple, pour transmettre `A` en morse on fera `=_===`.

1.1 Quelle sera la chaîne de caractère permettant de transmettre `N` en morse ?

1.2 Traduis le code suivant : `===_=_=_===`.

1.3 Quelle représentation utilisera-t-on pour l'espacement entre deux lettres ? pour l'espacement entre deux mots ?

1.4 Pour transmettre `BA BA` on fera `===_=_=_===___=_===_______===_=_=_===___=_===`.
Traduis le code suivant : `===___===_===_===_______===___===_===_===`

1.5 Comment faire pour transmettre `GA BU` ?

Exercice 2 *Écrire un convertisseur de lettre*
Commençons par écrire un convertisseur de lettre. Pour cela, on propose la classe `Lettre` qui permet de représenter une lettre ou le caractère "espace".

Lettre
- alphabetMorse:String [*]
- lettre: char
+ Lettre(lettre:char)
+ Lettre(morse:String)
+ toNumero():int
+ toChar():char
+ toMorse():String
+ toString():String

```
1  public class ExecutableMorse {
2      public static void main(String[] args) {
3          Lettre n = new Lettre('N');
4          assert n.toChar() == 'N';
5          assert n.toMorse().equals("===_=");
6          Lettre a = new Lettre("=_===");
7          assert a.toMorse().equals("=_===");
8          assert a.toChar() == 'A';
9    }
10 }
```

2.1 Dans les tests d'égalité, pourquoi utilise-t-on un `==` à la ligne 4 alors qu'on utilise la méthode `equals` à la ligne 5 ?

2.2 Ajoute des tests dans la classe exécutable.

2.3 Écris le code de la classe `Lettre`. N'oublie pas d'appliquer les *bonnes* méthodes :

- On commence par écrire du code qui compile mais qui ne fait RIEN
- On vérifie que les tests ECHOUENT
- On code les méthodes une par une en vérifiant à chaque étape que les tests concernés passent.

Quelques précisions :

- l'attribut statique `alphabetMorse` est une liste de chaînes de caractères modélisant les transmissions en morse des lettres de l'alphabet et du caractère "espace".
- la méthode privée `toNumero()` renvoie une valeur comprise entre 0 et 26 (par exemple, 0 pour `A`, 1 pour `B`, ..., 25 pour `Z` et 26 pour le caractère "espace")

Pour coder la méthode `toNumero()`, vous pouvez utiliser le code ascii des lettres. En java, un simple cast permet de récupérer le code ASCII d'un caractère.

```
char caractere = 'T';
int codeASCII = (int) caractere;
System.out.println(codeASCII);
```

Exercice 3 *Écrire un convertisseur de texte*
On veut maintenant écrire une classe `Texte` avec :

- un attribut de type `List<Lettre>`,
- un constructeur prenant en paramètre une chaîne (String) et construisant la liste de lettres correspondant à la chaîne.
- une méthode `toString()` qui renvoie le texte en clair,
- une méthode `toMorse()` qui renvoie le texte en morse.

3.1 Complète la classe exécutable de façon à ajouter des tests pour cette classe.

3.2 Écris le code de la classe `Texte` sans oublier de coder avec les *bonnes* méthodes.

Exercice 4 *Compléments*

4.1 On veut ajouter à la classe `Texte` la méthode `public boolean contient(Lettre lettre)` qui prend une lettre en paramètre et qui renvoie vrai si le texte contient la lettre.
Code cette méthode en utilisant la méthode `contains` de `ArrayList`. Tu auras besoin de modifier le code de la méthode `equals` de la classe `Lettre`.
Bien entendu, tu n'as pas oublié d'écrire des tests AVANT de coder cette méthode !

4.2 Ajoute à la classe `Texte` une méthode `public String decode(String texteEnMorse)` qui prend en paramètre un texte en morse et renvoie la chaîne correspondant au texte "décodé".
Bien entendu, tu n'as pas oublié d'écrire des tests AVANT de coder cette méthode !

4.3 Tu peux faire « du son » en utilisant la classe suivante. Ajoute une méthode `toSon()` à la classe `Texte` pour que le code morse soit « joué » sur la carte son.

```
import javax.sound.sampled.*;

public class Son{
  SourceDataLine sdl;
  Son(){
    try{
      sdl  = AudioSystem.getSourceDataLine(new AudioFormat(8000f, 8,1,true,
         false) );
      sdl.open(new AudioFormat(8000f, 8,1,true,false) );
      sdl.start();
      sdl.flush();
    }
    catch(LineUnavailableException e)
    {
    }
  }

  public void pause(){
    try{
      Thread.sleep(100);
    }
    catch( InterruptedException e){
    }
  }

  public void tone(int msecs)
  {
    byte[] buf = new byte[msecs*8];
    for (int i=0; i < msecs*8; i++) {
      double angle = i / (8000f / 440) * 2.0 * Math.PI;
      buf[i] = (byte)(Math.sin(angle) * 127.0);
    }
    sdl.write(buf,0,8*msecs);
    sdl.drain();
  }

  public static void main(String[] args) throws Exception {
    Son r = new Son();
    r.tone(100);
    r.pause();

  }
}
```