

Programmation Objet – Initiation à Java

Letícia SEIXAS PEREIRA

Cours 04 – La notion d'Objet

04/11/2019

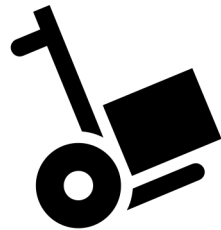
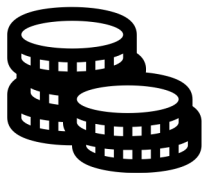


La notion d'Objet

- L'un des objectifs de la programmation objet est de simuler, à l'aide d'un programme informatique, la manipulation des objets réels par l'être humain;
- Tout comme nous manipulons les objets réels, les applications informatiques manipulent des objets virtuels.

La notion d'Objet

- L'un des objectifs de la programmation objet est de simuler, à l'aide d'un programme informatique, la manipulation des objets réels par l'être humain;
- Tout comme nous manipulons les objets réels, les applications informatiques manipulent des objets virtuels.



La notion d'Objet

: Regardez autour de vous et vous trouverez de nombreux exemples d'objets du monde réel:
votre **ordinateur**, votre **bureau**, votre **téléphone**, votre **sac à dos**....

La notion d'Objet

: Regardez autour de vous et vous trouverez de nombreux exemples d'objets du monde réel: votre **ordinateur**, votre **bureau**, votre **téléphone**, votre **sac à dos**....

- Les objets du monde réel partagent deux caractéristiques: Ils ont tous un **état** et un **comportement**.

La notion d'Objet

: Regardez autour de vous et vous trouverez de nombreux exemples d'objets du monde réel: votre **ordinateur**, votre **bureau**, votre **téléphone**, votre **sac à dos**....

- Les objets du monde réel partagent deux caractéristiques: Ils ont tous un **état** et un **comportement**.



- Les chiens ont l'**état** (nom, couleur, poids...) et le **comportement** (aboielements, aller chercher...).

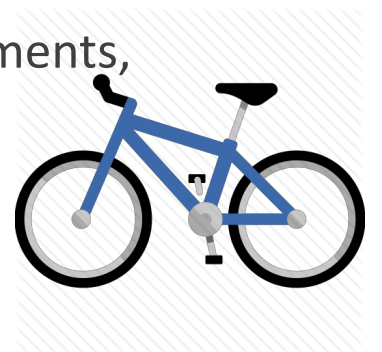
La notion d'Objet

: Regardez autour de vous et vous trouverez de nombreux exemples d'objets du monde réel: votre **ordinateur**, votre **bureau**, votre **téléphone**, votre **sac à dos**....

- Les objets du monde réel partagent deux caractéristiques: Ils ont tous un **état** et un **comportement**.



- Les chiens ont l'**état** (nom, couleur, poids...) et le **comportement** (aboielements, aller chercher...).



- Les bicyclettes ont aussi un **état** (vitesse actuelle, cadence actuelle de la pédale...) et le **comportement** (changer la vitesse, changer la cadence de la pédale, freiner...).

La notion d'Objet

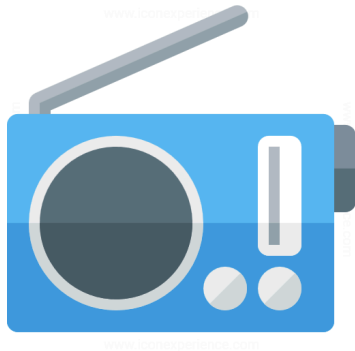
Pour chaque objet que vous voyez, posez-vous deux questions:

- «Quels états possibles cet objet peut-il avoir? »
- « Quel sont les comportements possibles de cet objet? »

La notion d'Objet

Pour chaque objet que vous voyez, posez-vous deux questions:

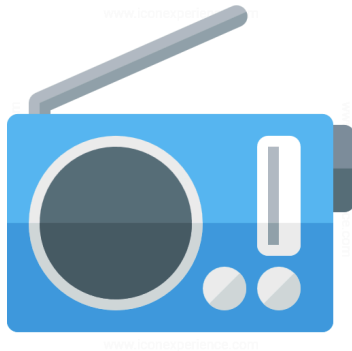
- «Quels états possibles cet objet peut-il avoir? »
- « Quel sont les comportements possibles de cet objet? »



La notion d'Objet

Pour chaque objet que vous voyez, posez-vous deux questions:

- «Quels états possibles cet objet peut-il avoir? »
- « Quel sont les comportements possibles de cet objet? »



États: on, off, volume actuel, station actuelle;

Comportement: allumer , éteindre, augmenter le volume, diminuer le volume, rechercher station...

La notion d'Objet

Pour chaque objet que vous voyez, posez-vous deux questions:

- «Quels états possibles cet objet peut-il avoir? »
- « Quel sont les comportements possibles de cet objet? »

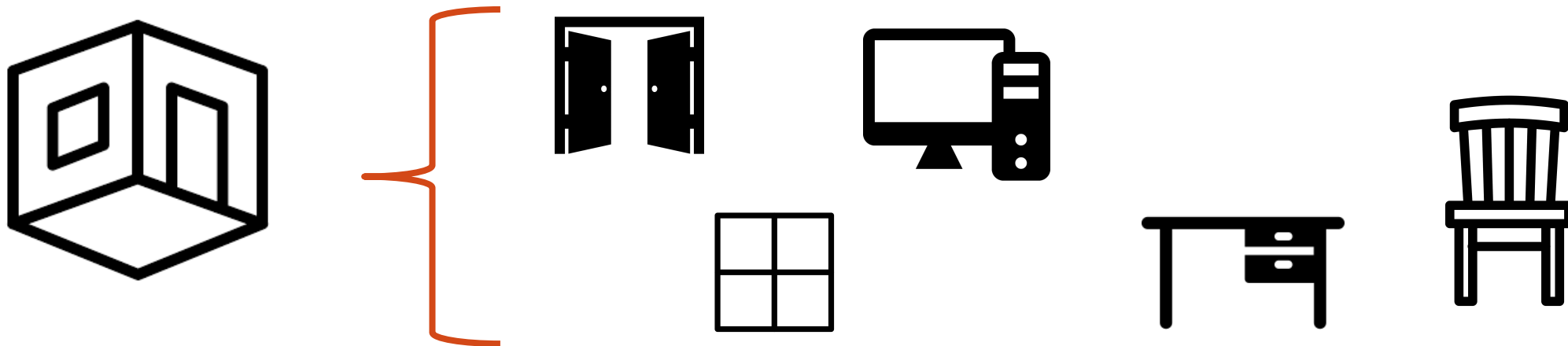
Certains objets, à leur tour, contiendront également d'autres objets:

La notion d'Objet

Pour chaque objet que vous voyez, posez-vous deux questions:

- «Quels états possibles cet objet peut-il avoir? »
- « Quel sont les comportements possibles de cet objet? »

Certains objets, à leur tour, contiendront également d'autres objets:



La notion d'Objet

Pour chaque objet que vous voyez, posez-vous deux questions:

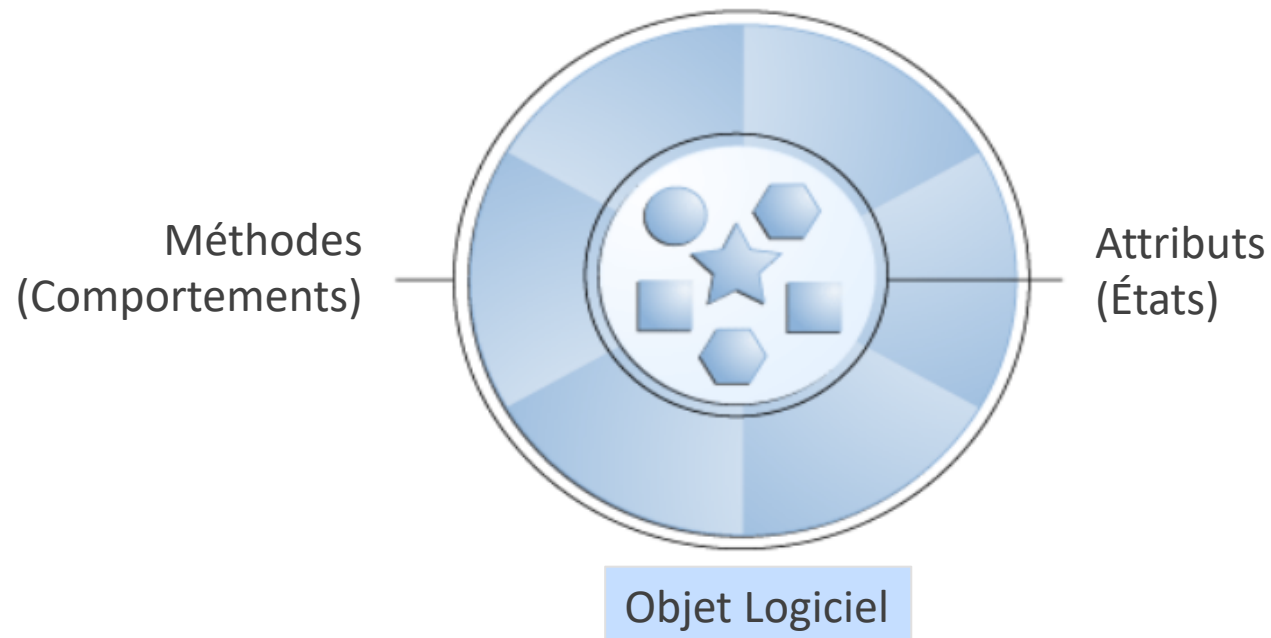
- «Quels états possibles cet objet peut-il avoir? »
- « Quel sont les comportements possibles de cet objet? »

Ces observations du monde réel se traduisent toutes dans le monde de la programmation orientée objet.

La notion d'Objet

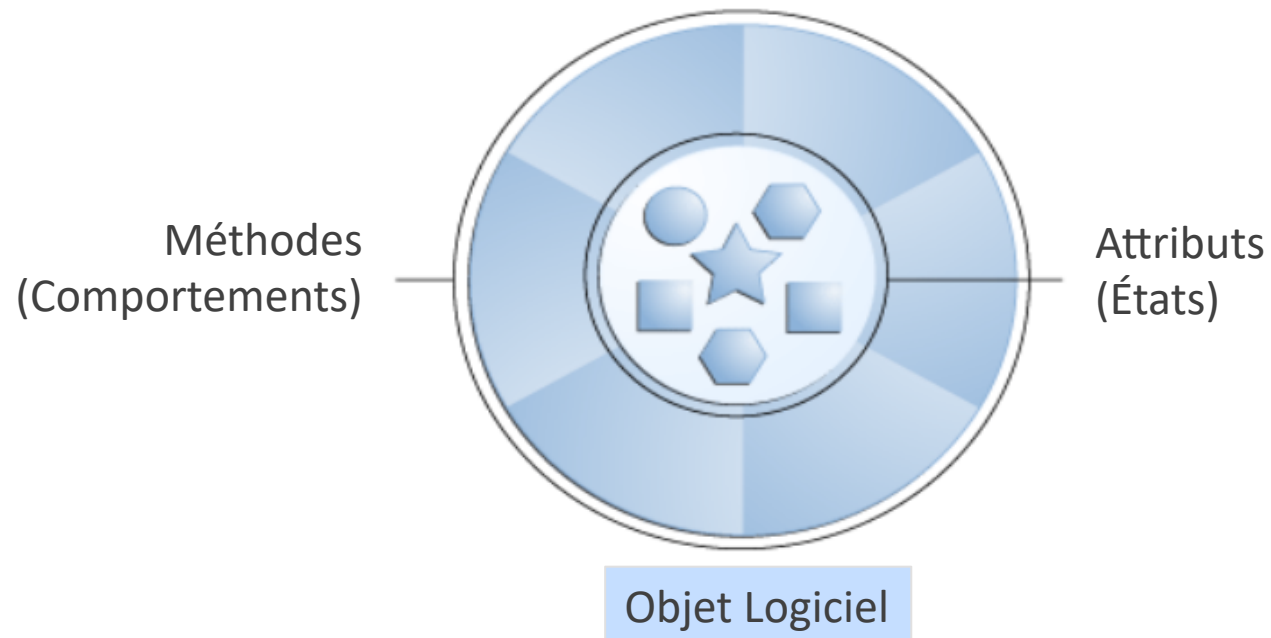
Les objets logiciels sont conceptuellement similaires aux objets du monde réel: ils sont eux aussi constitués d'un **état** et d'un **comportement** :

- *Un objet stocke son **état** dans des **attributs** et expose son **comportement** par des **méthodes**.*

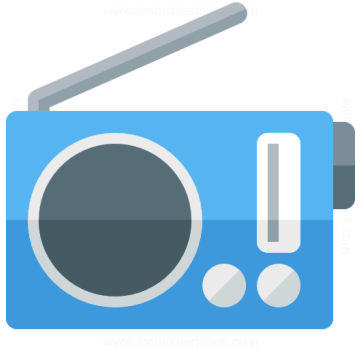


La notion d'Objet

- Les méthodes agissent sur l'état interne d'un objet et servent de mécanisme principal pour la communication objet-objet;



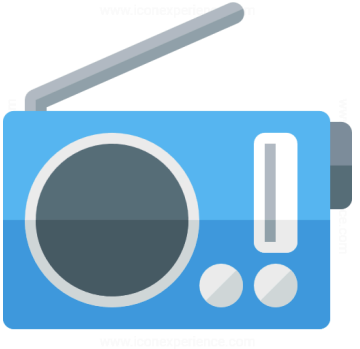
La notion d'Objet



États: on, off, volume actuel, station actuelle;

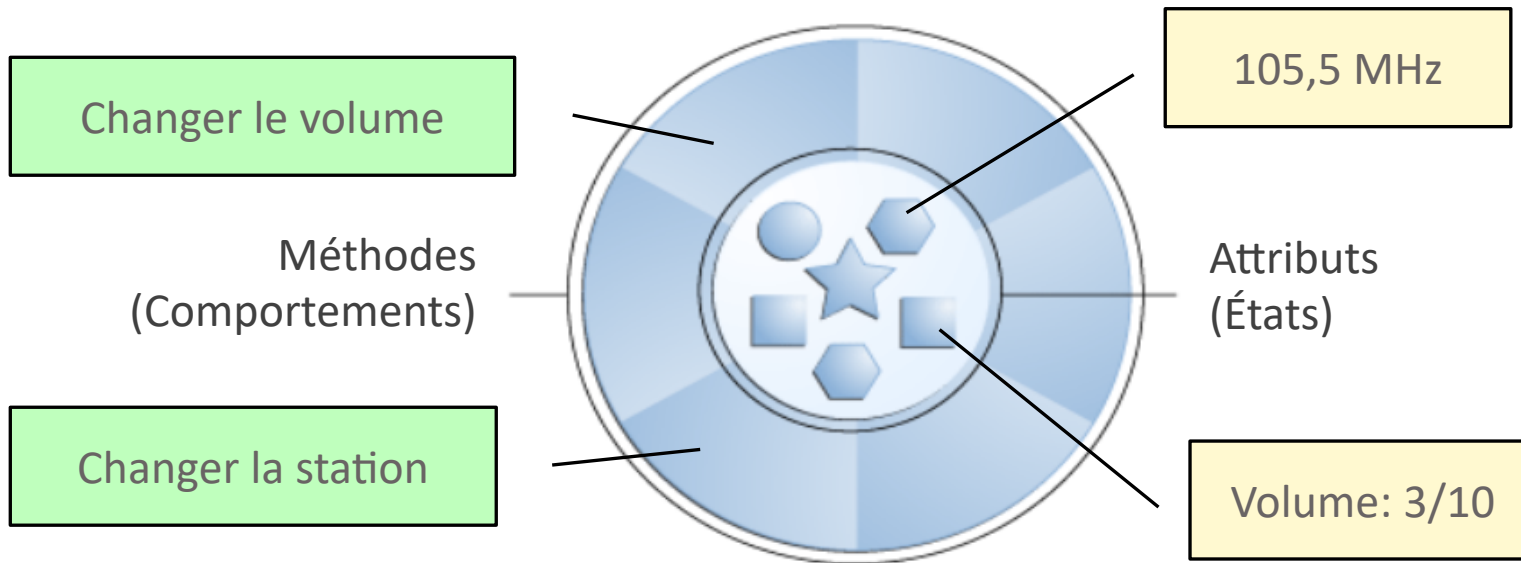
Comportement: allumer , éteindre, augmenter le volume, diminuer le volume, rechercher station.

La notion d'Objet



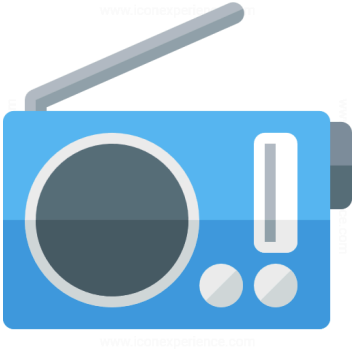
États: on, off, volume actuel, station actuelle/fréquence;

Comportement: allumer , éteindre, augmenter le volume, diminuer le volume, rechercher station.

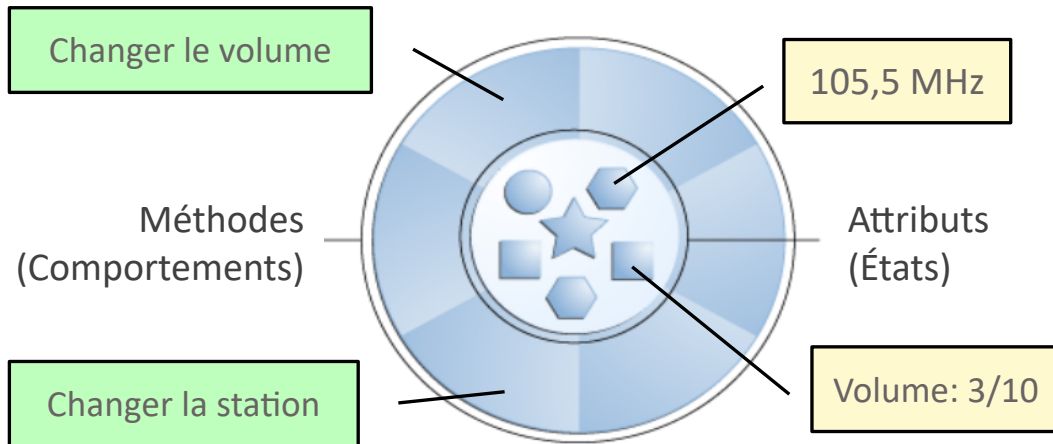


Une radio modélisée en tant qu'objet logiciel

La notion d'Objet

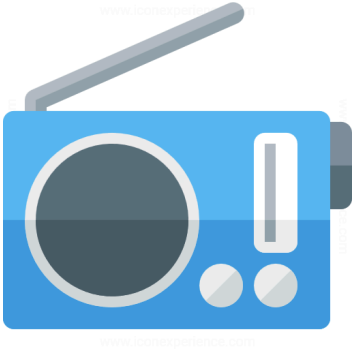


En attribuant un **état** (fréquence et volume) et en fournissant des **méthodes pour changer cet état**, l'objet reste dans le contrôle de la façon dont le monde extérieur est autorisé à l'utiliser:

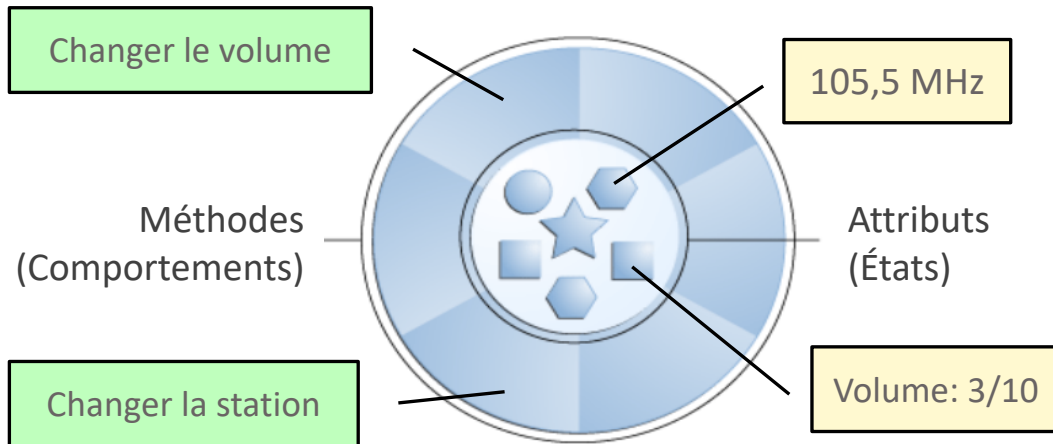


Une radio modélisée en tant qu'objet logiciel

La notion d'Objet



En attribuant un **état** (fréquence et volume) et en fournissant des **méthodes pour changer cet état**, l'objet reste dans le contrôle de la façon dont le monde extérieur est autorisé à l'utiliser:



Une radio modélisée en tant qu'objet logiciel

- Par exemple: si la radio n'a que 10 niveaux de volume, une méthode pour changer de volume pourrait rejeter toute valeur inférieure à 1 ou supérieure à 10.

La notion d'Objet

Le regroupement de code dans des **objets** logiciels individuels offre un certain nombre d'avantages, par ex:

- **Modularité:** Le code source d'un objet peut être écrit et maintenu indépendamment du code source des autres objets. Une fois créé, un objet peut être facilement transmis autour du système;

La notion d'Objet

Le regroupement de code dans des **objets** logiciels individuels offre un certain nombre d'avantages, par ex:

- **Modularité**: Le code source d'un objet peut être écrit et maintenu indépendamment du code source des autres objets. Une fois créé, un objet peut être facilement transmis autour du système;
- **Masquage des informations** : En interagissant uniquement avec les méthodes d'un objet, les détails de son implémentation interne restent cachés du monde extérieur;

La notion d'Objet

Le regroupement de code dans des **objets** logiciels individuels offre un certain nombre d'avantages, par ex:

- **Modularité**: Le code source d'un objet peut être écrit et maintenu indépendamment du code source des autres objets. Une fois créé, un objet peut être facilement transmis autour du système;
- **Masquage des informations** : En interagissant uniquement avec les méthodes d'un objet, les détails de son implémentation interne restent cachés du monde extérieur;
- **Réutilisation du code**: Si un objet existe déjà (peut-être écrit par un autre développeur ou pour une autre tâche), il est possible d'utiliser cet objet dans un autre programme;

La notion d'Objet

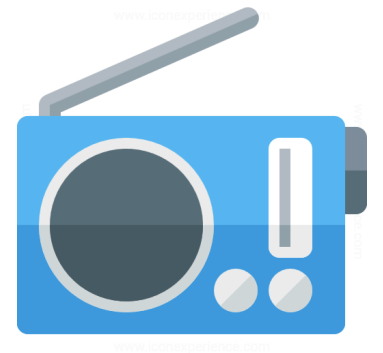
Le regroupement de code dans des **objets** logiciels individuels offre un certain nombre d'avantages, par ex:

- **Modularité:** Le code source d'un objet peut être écrit et maintenu indépendamment du code source des autres objets. Une fois créé, un objet peut être facilement transmis autour du système;
- **Masquage des informations :** En interagissant uniquement avec les méthodes d'un objet, les détails de son implémentation interne restent cachés du monde extérieur;
- **Réutilisation du code:** Si un objet existe déjà (peut-être écrit par un autre développeur ou pour une autre tâche), il est possible d'utiliser cet objet dans un autre programme;
- **Facilité d'enfichage et de débogage:** Si un objet particulier se révèle être problématique, il est possible de simplement le retirer de votre application et brancher un autre objet en remplacement. Ceci est analogue à la résolution de problèmes mécaniques dans le monde réel = si un boulon se casse, vous le remplacez, pas toute la machine.

Qu'est-ce qu'une classe?

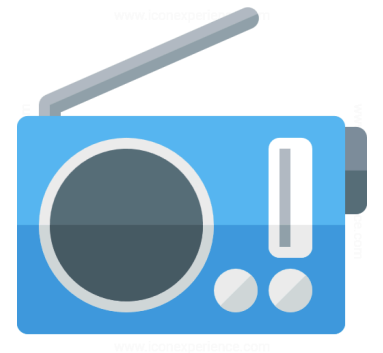
Qu'est-ce qu'une classe?

- Dans le monde réel, on trouve souvent de nombreux objets individuels tous du même genre;
- Il existe peut-être des milliers d'autres radios, toutes de même marque et modèle;



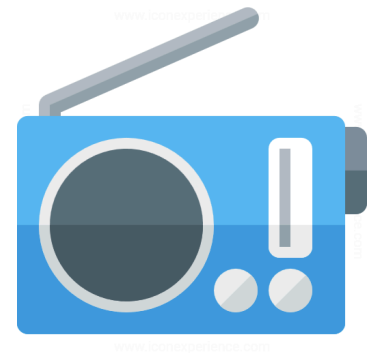
Qu'est-ce qu'une classe?

- Dans le monde réel, on trouve souvent de nombreux objets individuels tous du même genre;
- Il existe peut-être des milliers d'autres radios, toutes de même marque et modèle;
- Chaque radio a été construite à partir du même ensemble de **plans** et **contient donc les mêmes composants**;



Qu'est-ce qu'une classe?

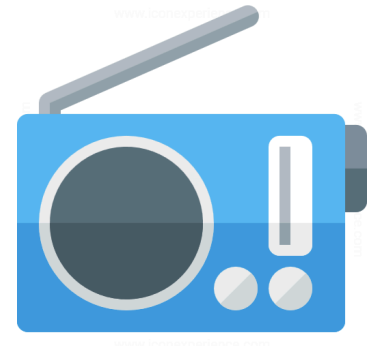
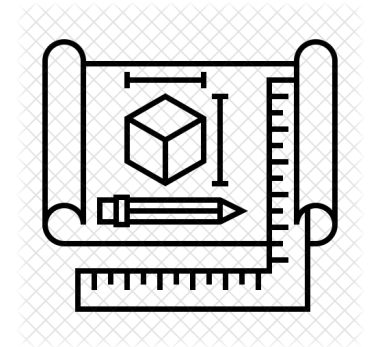
- Dans le monde réel, on trouve souvent de nombreux objets individuels tous du même genre;
- Il existe peut-être des milliers d'autres radios, toutes de même marque et modèle;
- Chaque radio a été construite à partir du même ensemble de **plans** et **contient donc les mêmes composants**;
- En termes orientés objet, on dit que **votre radio** est une **instance** de la **classe d'objets** connus sous le nom de **radios**;
- Une classe est le plan à partir duquel les objets individuels sont créés.



Qu'est-ce qu'une classe?

La classe Radio suivante est une implémentation possible d'une radio:

```
class Radio {  
  
    boolean on;  
    int volume;  
    double frequence;  
  
    void changerVolume (int nouveauVolume) { this.volume = nouveauVolume;}  
    void changerFrequence (double nouvelleFrequence) {this.frequence = nouvelleFrequence;}  
    void allumer() { this.on = true;}  
    void eteindre() { this.on = false;}  
    void afficherEtats() {  
        System.out.println("On/Off: " + (this.on ? "On" : "Off") +  
            " Volume: " + this.volume +  
            " Frequence: " + this.frequence );  
    }  
}
```



Qu'est-ce qu'une classe?

La classe Radio suivante est une implémentation possible d'une radio:

```
class Radio {
```

```
    boolean on;  
    int volume;  
    double frequence;
```

Les champs *on*, *volume* et *frequence* représentent l'état de l'objet

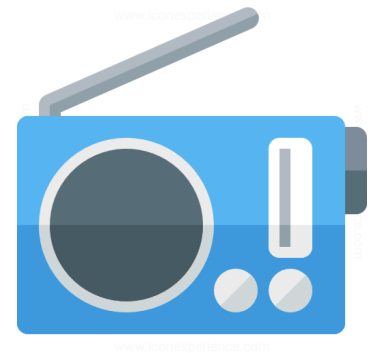
```
    void changerVolume (int nouveauVolume) { this.volume = nouveauVolume;}  
    void changerFrequence (double nouvelleFrequence) {this.frequence = nouvelleFrequence;}  
    void allumer() { this.on = true;}  
    void eteindre() { this.on = false;}  
    void afficherEtats() {  
        System.out.println("On/Off: " + (this.on ? "On" : "Off") +  
            " Volume: " + this.volume +  
            " Frequence: " + this.frequence );  
    }  
}
```

Les méthodes (*changerVolume*, *changerFrequence*, *Allumer*, etc.) définissent son interaction avec le monde extérieur.



Qu'est-ce qu'une classe?

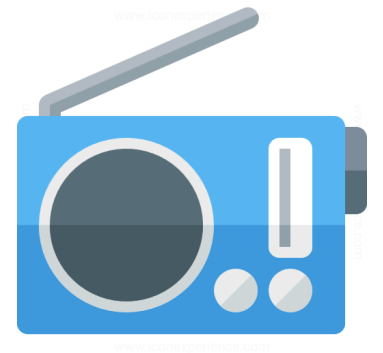
La classe Radio ne contient pas de méthode principale:



Qu'est-ce qu'une classe?

La classe Radio ne contient pas de méthode principale:

- Parce que ce n'est pas une application complète = c'est juste le plan pour les radios qui pourraient être utilisés dans une application;
- La responsabilité de créer et d'utiliser de nouveaux objets Radio appartient à une autre classe d'une application.



Qu'est-ce qu'une classe?

La classe RadioDemo suivante crée deux objets Radio distincts et appelle leurs méthodes:

```
class RadioDemo {  
  
    public static void main(String[] args) {  
  
        //Créer deux objets radios différents  
        Radio radio1 = new Radio();  
        Radio radio2 = new Radio();  
  
        //Invoquer des méthodes sur ces objets  
        radio1.allumer();  
        radio1.changerFrequence(107.5);  
        radio1.changerVolume(5);  
        radio1.afficherEtats();  
  
        radio2.allumer();  
        radio2.changerFrequence(105.1);  
        radio2.changerVolume(3);  
        radio2.afficherEtats();  
  
    }  
}
```



Qu'est-ce qu'une classe?

La classe RadioDemo suivante crée deux objets Radio distincts et appelle leurs méthodes:

```
class RadioDemo {  
  
    public static void main(String[] args) {  
  
        //Créer deux objets radios différents  
        Radio radio1 = new Radio();  
        Radio radio2 = new Radio();  
  
        //Invoquer des méthodes sur ces objets  
        radio1.allumer();  
        radio1.changerFrequence(107.5);  
        radio1.changerVolume(5);  
        radio1.afficherEtats();  
  
        radio2.allumer();  
        radio2.changerFrequence(105.1);  
        radio2.changerVolume(3);  
        radio2.afficherEtats();  
  
    }  
}
```



On/Off: On Volume: 5 Frequence: 107.5
On/Off: On Volume: 3 Frequence: 105.1



Éléments de syntaxe

Type de données

- Une variable est un élément qui stocke des informations de toute sorte en mémoire;
- Les types de variables en Java sont répartis en deux catégories:
 - des variables de type simple ou « primitif »;
 - **des variables de type structurés.**
- Déclaration d'une variable:
 - `<type de la variable> <nom de la variable>`

```
int x;  
  
int y = 2;  
  
int age = 15;  
  
int qtEtudiants;  
  
int qtEtudiantsM1;
```

Les objets en java

- L'idée de base de la programmation orientée objet est de rassembler dans une même entité appelée **objet** les **données** et les **traitements** qui s'y appliquent.

Les objets en java

- L'idée de base de la programmation orientée objet est de rassembler dans une même entité appelée **objet** les **données** et les **traitements** qui s'y appliquent.
- *Comment créer un objet?*

Les objets en java

- L'idée de base de la programmation orientée objet est de rassembler dans une même entité appelée **objet** les **données** et les **traitements** qui s'y appliquent.
- *Comment créer un objet?*
 - Il faut définir la structure de l'objet.

Les objets en java

- Un objet ou **instance** est une variable *presque* comme les autres. Il faut notamment qu'il soit déclaré avec son type.

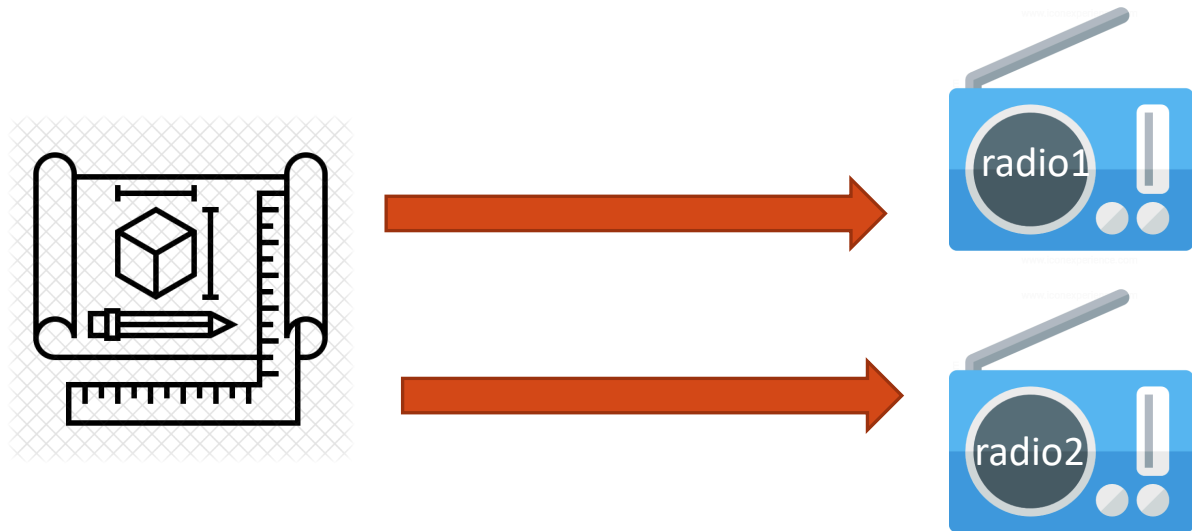
Les objets en java

- Un objet ou **instance** est une variable *presque* comme les autres. Il faut notamment qu'il soit déclaré avec son type.
- Le type d'un objet est un type complexe (par opposition aux types primitifs: entiers, caractères...) qu'on appelle une **classe**.

Constructeurs

Constructeurs

- Une classe contient des constructeurs qui sont appelés pour créer des objets à partir du ***plan de classe***.



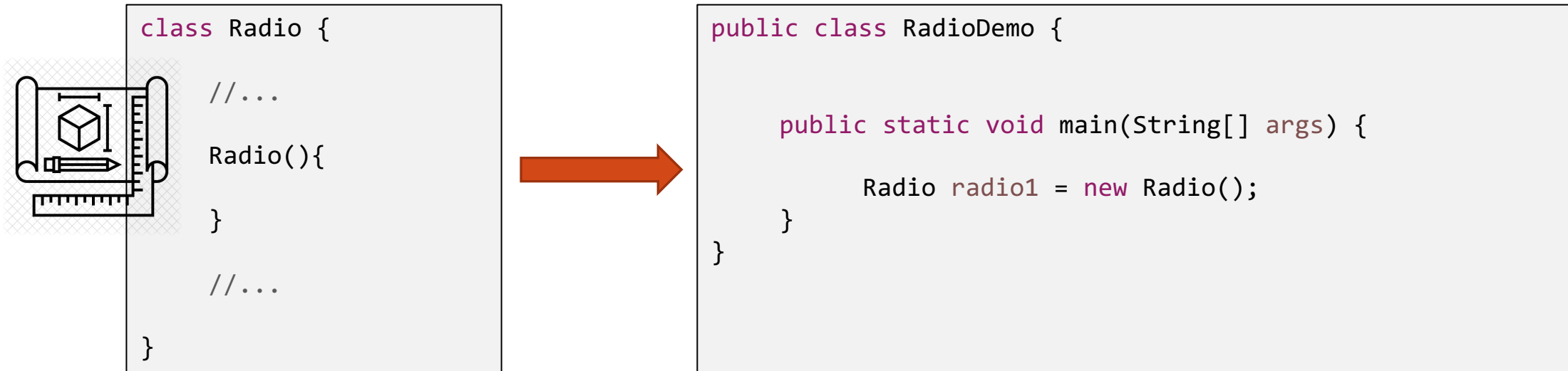
Constructeurs

- Les déclarations des constructeurs ressemblent à des déclarations de méthode, sauf qu'elles utilisent **le nom de la classe** et **n'ont pas de type de retour**:

```
class Radio {  
    //...  
    Radio(){  
    }  
    //...  
}
```

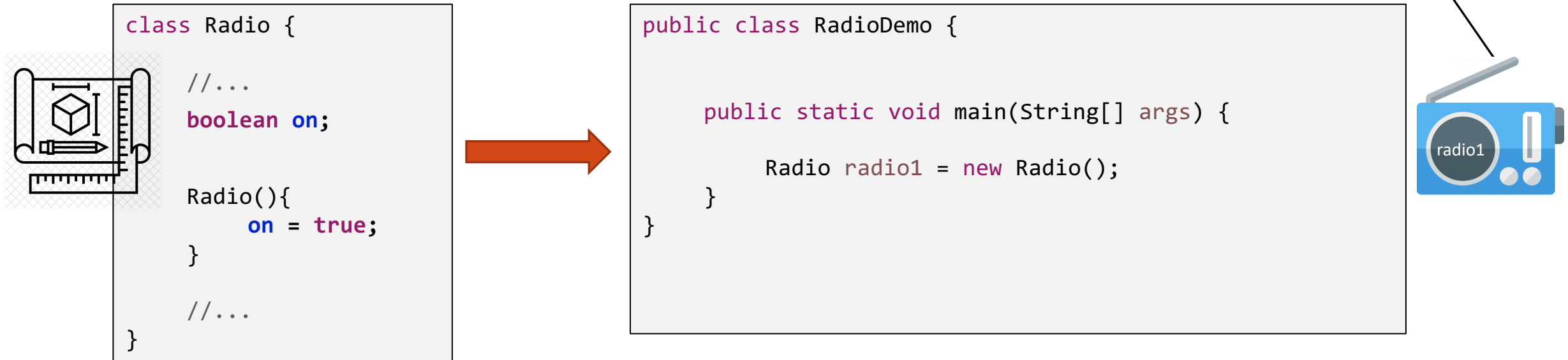
Constructeurs

- Les déclarations des constructeurs ressemblent à des déclarations de méthode, sauf qu'elles utilisent **le nom de la classe** et **n'ont pas de type de retour**;
- Pour créer un nouvel objet **Radio** appelé **radio1**, un constructeur est appelé par l'opérateur **new**:
 - **new Radio()** : L'opérateur **new** crée un espace dans la mémoire pour l'objet et initialise ses attributs.



Constructeurs

- Les déclarations des constructeurs ressemblent à des déclarations de méthode, sauf qu'elles utilisent **le nom de la classe** et **n'ont pas de type de retour**;
- Pour créer un nouvel objet **Radio** appelé **radio1**, un constructeur est appelé par l'opérateur **new**:
 - **new Radio()** : L'opérateur **new** crée un espace dans la mémoire pour l'objet et initialise ses attributs.

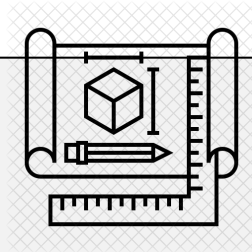


Constructeurs

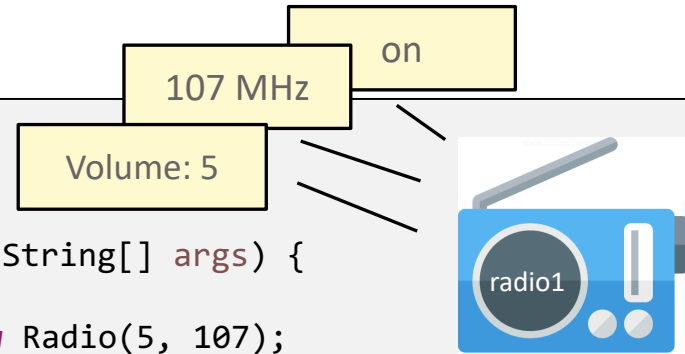
- Les déclarations des constructeurs ressemblent à des déclarations de méthode, sauf qu'elles utilisent **le nom de la classe** et **n'ont pas de type de retour**;
- Pour créer un nouvel objet **Radio** appelé **radio1**, un constructeur est appelé par l'opérateur **new**:
 - **new Radio()** : L'opérateur **new** crée un espace dans la mémoire pour l'objet et initialise ses attributs.

```
class Radio {  
  
    boolean on;  
    int volume;  
    double frequency;
```

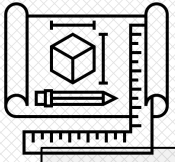
```
    Radio(int newVolume, double newFrequency){  
        on = true;  
        volume = newVolume;  
        frequency = newFrequency;  
    }  
    //...  
}
```



```
public class RadioDemo {  
  
    public static void main(String[] args) {  
        Radio radio1 = new Radio(5, 107);  
    }  
}
```



Constructeurs



```
class Radio {  
  
    boolean on;  
    int volume;  
    double frequency;  
  
    Radio(){  
        on = true;  
    }  
    Radio(int newVolume, double newFrequency){  
        volume = newVolume;  
        frequency = newFrequency;  
    }  
    //...  
}
```

- Il était possible de déclarer les deux constructeurs dans la classe **Radio** parce qu'ils ont des différents arguments;
- Comme pour les méthodes, la plateforme Java différencie les constructeurs en fonction du **nombre d'arguments** dans la liste et de **leurs types**;
- Il **n'est pas possible** d'écrire deux constructeurs qui ont le **même nombre** et le **même type d'arguments** pour la même classe.
 - *(La plate-forme ne serait pas capable de les différencier)*

Le mot-clé **this**

- Dans une méthode *d'instance* ou un constructeur, le mot-clé **this** est une référence à l'**objet courant** : l'objet dont la méthode ou le constructeur est appelé;
- Il est possible de faire une référence à n'importe quel membre de l'objet actuel à partir d'une méthode d'instance ou d'un constructeur en utilisant **this**.

Le mot-clé this

- L'utilisation de **this** avec un attribut:
 - Il est utilisé pour rendre le code **explicite** et **non ambigu**.

```
class Radio {  
  
    boolean on;  
    int volume;  
    double frequency;  
  
    Radio(int volume, double frequency){  
        on = true;  
        volume = volume;  
        frequency = frequency;  
    }  
    //...  
}
```



```
class Radio {  
  
    boolean on;  
    int volume;  
    double frequency;  
  
    Radio(int volume, double frequency){  
        this.on = true;  
        this.volume = volume;  
        this.frequency = frequency;  
    }  
    //...  
}
```

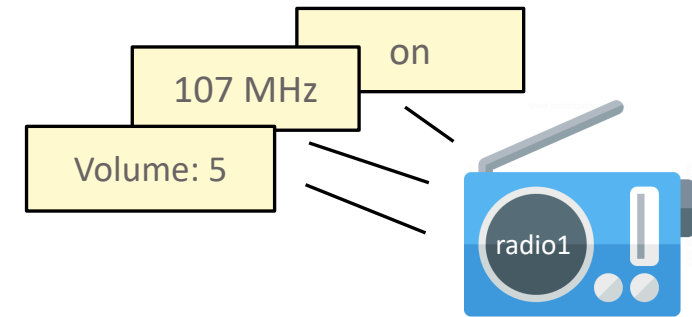

Le mot-clé this

- L'utilisation de **this** avec un attribut:
 - Il est utilisé pour rendre le code **explicite** et **non ambigu**.

```
class Radio {  
  
    boolean on;  
    int volume;  
    double frequency;  
  
    Radio(int volume, double frequency){  
        this.on = true;  
        this.volume = volume;  
        this.frequency = frequency;  
    }  
    //...  
}
```



```
public class RadioDemo {  
  
    public static void main(String[] args) {  
        Radio radio1 = new Radio (5, 107);  
    }  
}
```



Le mot-clé **this**

- L'utilisation de **this** avec un constructeur:
 - Depuis un constructeur, il est possible d'utiliser le **this** pour **appeler un autre constructeur** dans la même classe, c'est-à-dire, faire un ***appel explicite***:

Le mot-clé this

- L'utilisation de **this** avec un constructeur:
 - Depuis un constructeur, il est possible d'utiliser le **this** pour **appeler un autre constructeur** dans la même classe, c'est-à-dire, faire un ***appel explicite***:

```
class Radio {  
    //... attributs  
  
    Radio(){  
        this.on = true;  
    }  
    Radio(int volume, double frequency){  
        this.on = true;  
        this.volume = volume;  
        this.frequency = frequency;  
    }  
    Radio(boolean on, int volume, double frequency){  
        this.on = on;  
        this.volume = volume;  
        this.frequency = frequency;  
    }  
  
    //...  
}
```

Le mot-clé this

- L'utilisation de **this** avec un constructeur:
 - Depuis un constructeur, il est possible d'utiliser le **this** pour **appeler un autre constructeur** dans la même classe, c'est-à-dire, faire un *appel explicite*:

```
class Radio {  
    //... attributs  
  
    Radio(){  
        this.on = true;  
    }  
    Radio(int volume, double frequency){  
        this.on = true;  
        this.volume = volume;  
        this.frequency = frequency;  
    }  
    Radio(boolean on, int volume, double frequency){  
        this.on = on;  
        this.volume = volume;  
        this.frequency = frequency;  
    }  
  
    //...  
}
```



```
class Radio {  
    //... attributs  
  
    Radio(){  
        this(true, 0,0);  
    }  
    Radio (int volume, double frequency){  
        this(true, volume, frequency);  
    }  
    Radio (boolean on, int volume, double frequency){  
        this.on = on;  
        this.volume = volume;  
        this.frequency = frequency;  
    }  
  
    //...  
}
```

Le mot-clé this

```
class Radio {  
  
    //... attributs  
  
    Radio(){  
        this(true, 0,0);  
    }  
    Radio(int volume, double frequency){  
        this(true, volume, frequency);  
    }  
    Radio(boolean on, int volume, double frequency){  
        this.on = on;  
        this.volume = volume;  
        this.frequency = frequency;  
    }  
    //...  
}
```

- Chaque constructeur initialise **certaines ou toutes les attributs** membres d'une Radio;
- Les constructeurs fournissent une valeur par défaut pour chaque attribut membre dont la valeur initiale n'est pas fournie par un argument;
- Le compilateur détermine quel constructeur appeler, en fonction du **nombre** et du **type d'arguments**.

Le mot-clé this

```
class Radio {  
  
    //... attributs  
  
    Radio(){  
        this(true, 0,0);  
    }  
    Radio(int volume, double frequency){  
        this(true, volume, frequency);  
    }  
    Radio(boolean on, int volume, double frequency){  
        this.on = on;  
        this.volume = volume;  
        this.frequency = frequency;  
    }  
    //...  
}
```

- Chaque constructeur initialise **certaines ou toutes les attributs** membres d'une Radio;
- Les constructeurs fournissent une valeur par défaut pour chaque attribut membre dont la valeur initiale n'est pas fournie par un argument;
- Le compilateur détermine quel constructeur appeler, en fonction du **nombre** et du **type d'arguments**.

L'appel à un autre constructeur doit être sur la première ligne de cet autre constructeur.

La notion d'Objet

```
public class Rectangle {
```

```
    int longueur ;
```

```
    int largeur ;
```

```
    int surface () {
```

```
        return longueur * largeur ;
```

```
    }
```


```
}
```

- Un ensemble de **données** dites **attributs**;

- Un ensemble de **traitements** de ces données dites **méthodes**.

La notion d'Objet

```
public class Rectangle {  
    int longueur ;  
    int largeur ;  
  
    int surface () {  
        return longueur * largeur ;  
    }  
}
```



Rectangle
- longueur
- largeur
- surface

Les constructeurs

Exercice d'application : Comment peut-on appeler la méthode surface?

```
public class Rectangle {  
    int longueur ;  
    int largeur ;  
    public Rectangle(int longueur, int largeur){  
        this.longueur = longueur;  
        this.largeur = largeur;  
    }  
    int surface () {  
        return this.longueur * this.largeur ;  
    }  
}
```

Les constructeurs

Exercice d'application : Comment peut-on appeler la méthode surface?

```
public class Rectangle {  
  
    int longueur ;  
  
    int largeur ;  
  
    public Rectangle(int longueur, int largeur){  
  
        this.longueur = longueur;  
  
        this.largeur = largeur;  
  
    }  
  
    int surface () {  
  
        return this.longueur * this.largeur ;  
  
    }  
  
}
```

```
public class TestRectangle {  
  
    public static void main (String[] args) {  
  
        Rectangle rectangle1;  
  
        rectangle1 = new Rectangle(3, 4);  
  
        int surface1 = rectangle1.surface();  
  
        System.out.println(surface1);  
  
    }  
  
}
```

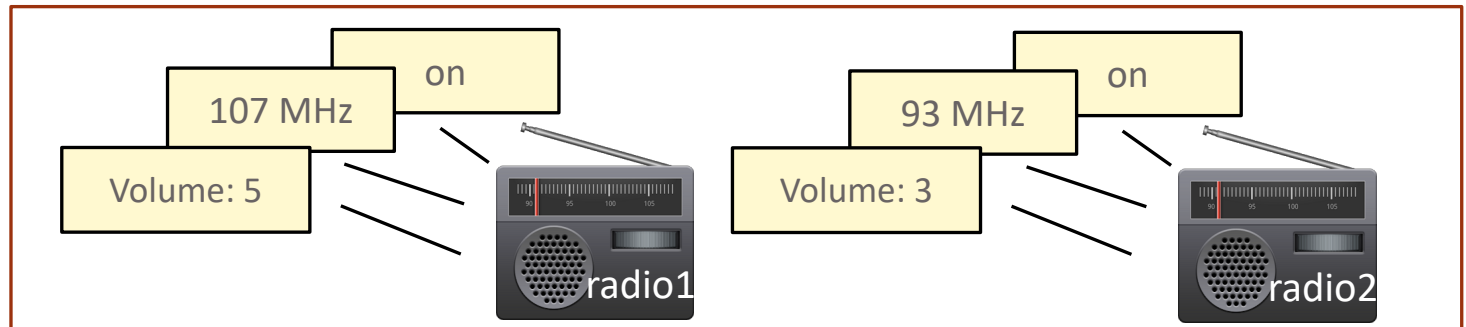
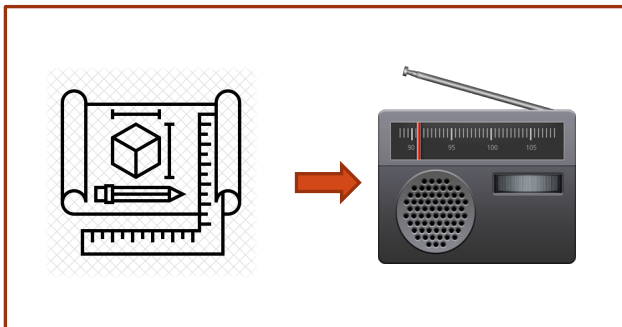
Membres de Classe

Membres de Classe

- Lorsqu'un certain nombre d'objets sont créés à partir du même plan de classe, ils ont chacun leurs propres copies distinctes des attributs d'instance;

Membres de Classe

- Dans le cas de la classe **Radio**, les variables d'instance sont « on », volume et fréquence:
 - Chaque objet **Radio** a ses propres valeurs pour ces variables, stockées dans différents emplacements de mémoire.



```
class Radio {  
  
    boolean on;  
    int volume;  
    double frequence;  
  
    //...  
}
```

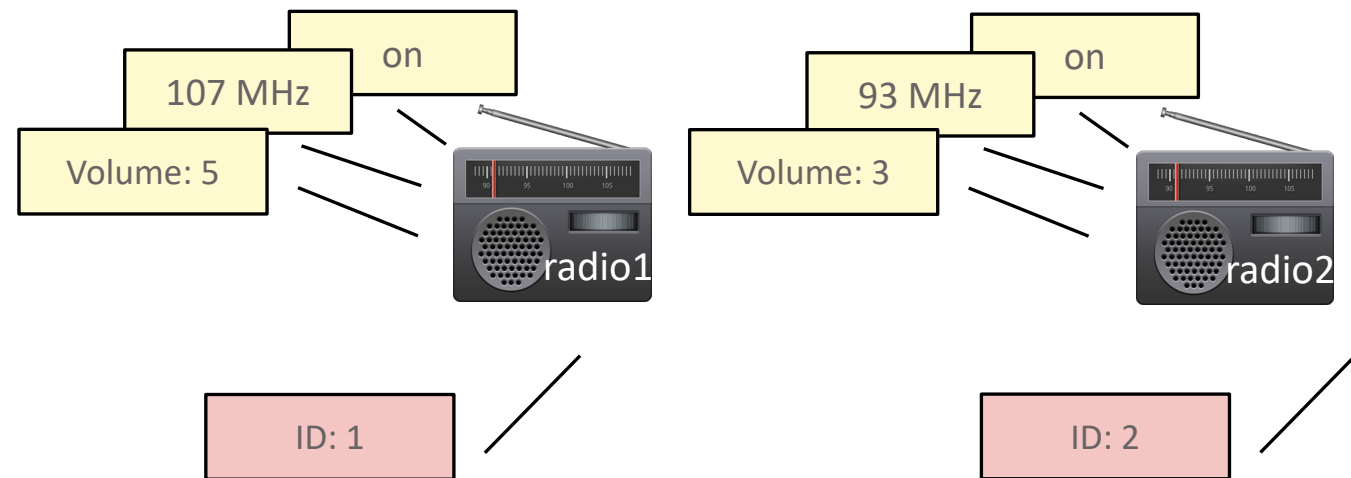
```
public class RadioDemo {  
  
    public static void main(String[] args) {  
  
        Radio radio1 = new Radio (5, 107);  
        Radio radio2 = new Radio (3, 93);  
  
    }  
}
```

Attributs de Classe

- Il est possible d'avoir des variables communes à tous les objets en utilisant le modificateur ***static*** (statique):
 - Les attributs qui ont le modificateur ***static*** dans leur déclaration sont appelés **attributs statiques** ou **attributs de classe**;
 - Ils sont associés à la classe, plutôt qu'avec n'importe quel objet;
 - Chaque instance de la classe partage les **attributs de classe**, qui se trouvent dans un **emplacement fixe en mémoire**;
- Tout objet instance de cette classe peut changer ces attributs, mais **ils peuvent également être manipulés sans créer une instance de cette classe**.

Attributs de Classe

- *Par exemple:*
 - Supposons que chaque objet Radio créé a un numéro de série commençant par 1 pour le premier objet;
 - Ce numéro d'identification est unique à chaque objet et est donc un **attribut d'instance**;



Attributs de Classe

- *Par exemple:*
 - Supposons que chaque objet Radio créé a un numéro de série commençant par 1 pour le premier objet;
 - Ce numéro d'identification est unique à chaque objet et est donc un **attribut d'instance**;
 - En même temps, nous avons besoin d'un attribut pour suivre le nombre d'objets Radio créés afin de savoir quel ID attribuer au suivant objet;
 - Cet attribut n'est lié à aucun objet individuel, mais à la classe dans son ensemble. Pour cela, il est nécessaire l'utilisation d'un **attribut de classe** (nombreRadios, par exemple).

Attributs de Classe

Attribut d'instance

Attribut de classe

```
class Radio {  
    boolean on;  
    int volume;  
    double frequence;  
    //Ajoute un attribut d'instance pour l'ID de l'objet  
    int id;  
    //Ajoute un attribut de classe pour le nombre d'objets Radio instanciés  
    static int nombreRadios = 0;  
    //...  
}
```

Attributs de Classe

Attribut d'instance

Attribut de classe

```
class Radio {  
    boolean on;  
    int volume;  
    double frequence;  
    //Ajoute un attribut d'instance pour l'ID de l'objet  
    int id;  
    //Ajoute un attribut de classe pour le nombre d'objets Radio instanciés  
    static int nombreRadios = 0;  
    //...  
}
```

```
public class RadioDemo {  
    public static void main(String[] args) {  
        Radio radio1 = new Radio (5, 107);  
        Radio radio2 = new Radio (3, 93);  
        System.out.println(Radio.nombreRadios);  
    }  
}
```

Attributs de Classe

Attribut d'instance

Attribut de classe

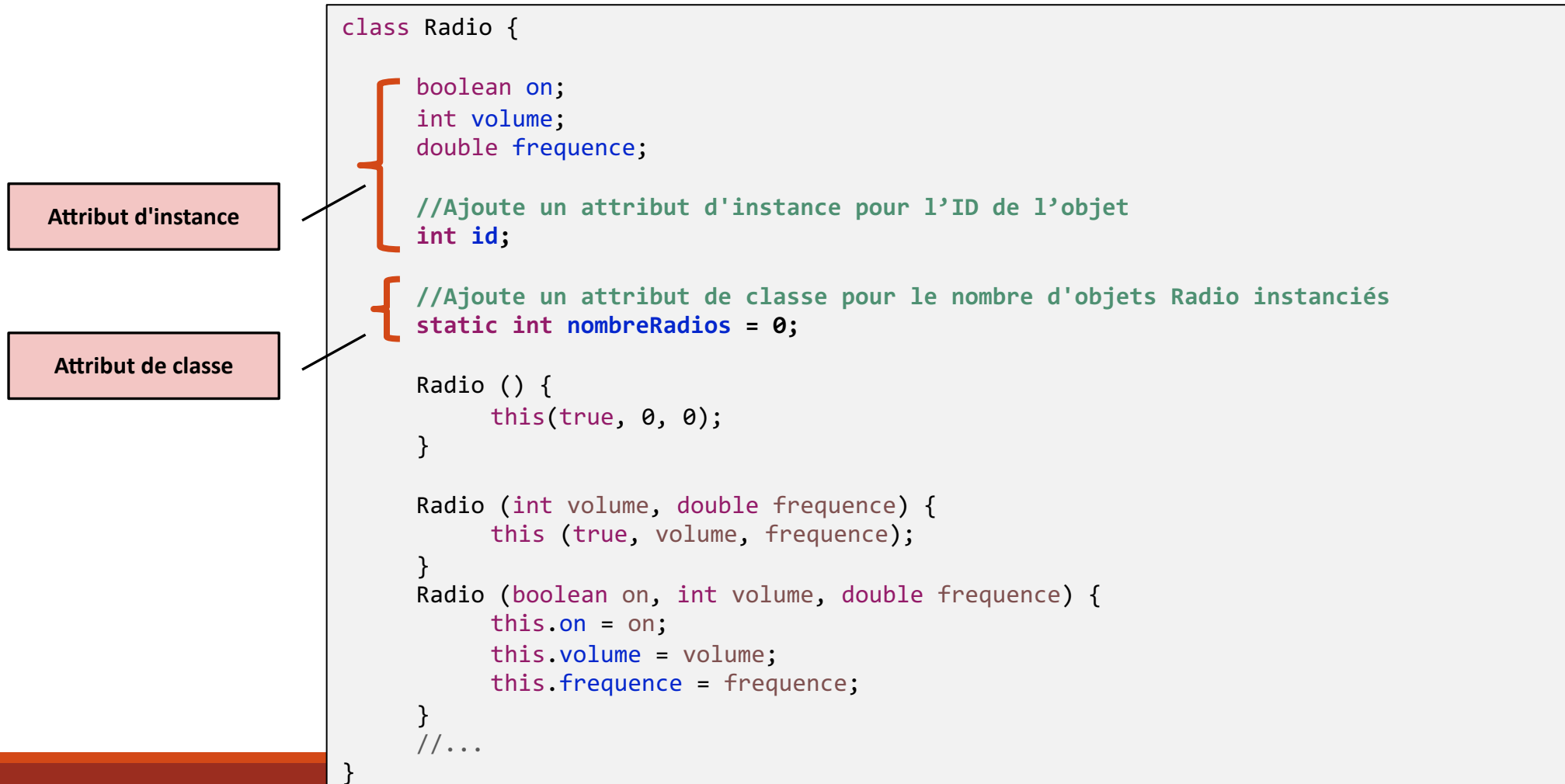
```
class Radio {  
    boolean on;  
    int volume;  
    double frequency;  
    //Ajoute un attribut d'instance pour l'ID de l'objet  
    int id;  
    //Ajoute un attribut de classe pour le nombre d'objets Radio instanciés  
    static int nombreRadios = 0;  
    //...  
}
```

```
public class RadioDemo {  
    public static void main(String[] args) {  
        Radio radio1 = new Radio (5, 107);  
        Radio radio2 = new Radio (3, 93);  
        System.out.println(Radio.nombreRadios);  
    }  
}
```

On/Off: On Volume: 5 Frequency: 107
On/Off: On Volume: 3 Frequency: 93
0

Attributs de Classe

Il est possible d'utiliser le constructeur de la classe, dans ce cas, *Radio*, pour définir l'attribut d'instance *id* et incrémenter l'attribut de classe *nombreRadios*:



Attributs de Classe

Il est possible d'utiliser le constructeur de la classe, dans ce cas, *Radio*, pour définir l'attribut d'instance *id* et incrémenter l'attribut de classe *nombreRadios*:

Attribut d'instance

Attribut de classe

```
class Radio {  
    boolean on;  
    int volume;  
    double frequence;  
  
    //Ajoute un attribut d'instance pour l'ID de l'objet  
    int id;  
  
    //Ajoute un attribut de classe pour le nombre d'objets Radio instanciés  
    static int nombreRadios = 0;  
  
    Radio () {  
        this(true, 0, 0);  
    }  
  
    Radio (int volume, double frequence) {  
        this (true, volume, frequence);  
    }  
  
    Radio (boolean on, int volume, double frequence) {  
        this.on = on;  
        this.volume = volume;  
        this.frequence = frequence;  
        nombreRadios = nombreRadios+ 1;  
        this.id = nombreRadios;  
    }  
    //...  
}
```

Attributs de Classe

Il est possible d'utiliser le constructeur de la classe, dans ce cas, *Radio*, pour définir l'attribut d'instance *id* et incrémenter l'attribut de classe *nombreRadios*:

Attribut d'instance

Attribut de classe


```
class Radio {  
    boolean on;  
    int volume;  
    double frequence;  
  
    //Ajoute un attribut d'instance pour l'ID de l'objet  
    int id;  
  
    //Ajoute un attribut de classe pour le nombre d'objets Radio instanciés  
    static int nombreRadios = 0;  
  
    Radio () {  
        this(true, 0, 0);  
    }  
  
    Radio (int volume, double frequence) {  
        this (true, volume, frequence);  
    }  
  
    Radio (boolean on, int volume, double frequence) {  
        this.on = on;  
        this.volume = volume;  
        this.frequence = frequence;  
        this.id= ++nombreRadios;  
    }  
    //...  
}
```

Attributs de Classe

Il est possible d'utiliser le constructeur de la classe, dans ce cas, **Radio**, pour définir l'attribut d'instance **id** et incrémenter l'attribut de classe **nombreRadios**:

```
class Radio {  
  
    boolean on;  
    int volume;  
    double frequence;  
  
    //Ajoute un attribut d'instance pour l'ID de l'objet  
    int id;  
  
    //Ajoute un attribut de classe pour le nombre d'objets Radio instanciés  
    static int nombreRadios = 0;  
  
    Radio() {  
        this(true, 0, 0);  
    }  
  
    Radio(int volume, double frequence) {  
        this(true, volume, frequence);  
    }  
  
    Radio(boolean on, int volume, double frequence) {  
        this.on = on;  
        this.volume = volume;  
        this.frequence = frequence;  
        this.id= ++nombreRadios;  
    }  
    //...  
}
```

```
public class RadioDemo {  
  
    public static void main(String[] args) {  
  
        Radio radio1 = new Radio(5, 107);  
        Radio radio2 = new Radio(3, 93);  
  
        radio1.afficherEtats();  
        radio2.afficherEtats();  
  
        System.out.println(Radio.nombreRadios);  
  
    }  
}
```



```
On/Off: On Volume: 5 Frequency: 107.0  
On/Off: On Volume: 3 Frequency: 93.0  
2
```

La notion de constante

- Les constantes sont des données visibles par toutes les méthodes de l'application:
 - Ces données sont déclarées en mode public;
 - Habituellement utilisé pour les données qui ne sont pas modifiables:

Ex:

```
static final double PI= 3.14159;
```


La notion d'objet

Exercice d'application: Créer une classe *Cercle* et une classe *TestCercle*:

1. Écrivez une classe **Cercle** dans un fichier Cercle.java ayant comme attribut le rayon du cercle (de type int) et une variable de classe (constant) PI ;
2. La classe **Cercle** offrira un constructeur prenant comme argument le rayon du cercle;
3. Écrivez une classe TestCercle dans un fichier TestCercle.java dans laquelle vous définirez un *main* afin de tester votre classe Cercle;
4. Définissez ensuite la méthode suivante dans la classe Cercle et testez votre programme.
*surface d'un cercle = PI * rayon * rayon (pi fois le carré du rayon)*

Cercle
- rayon
- surface