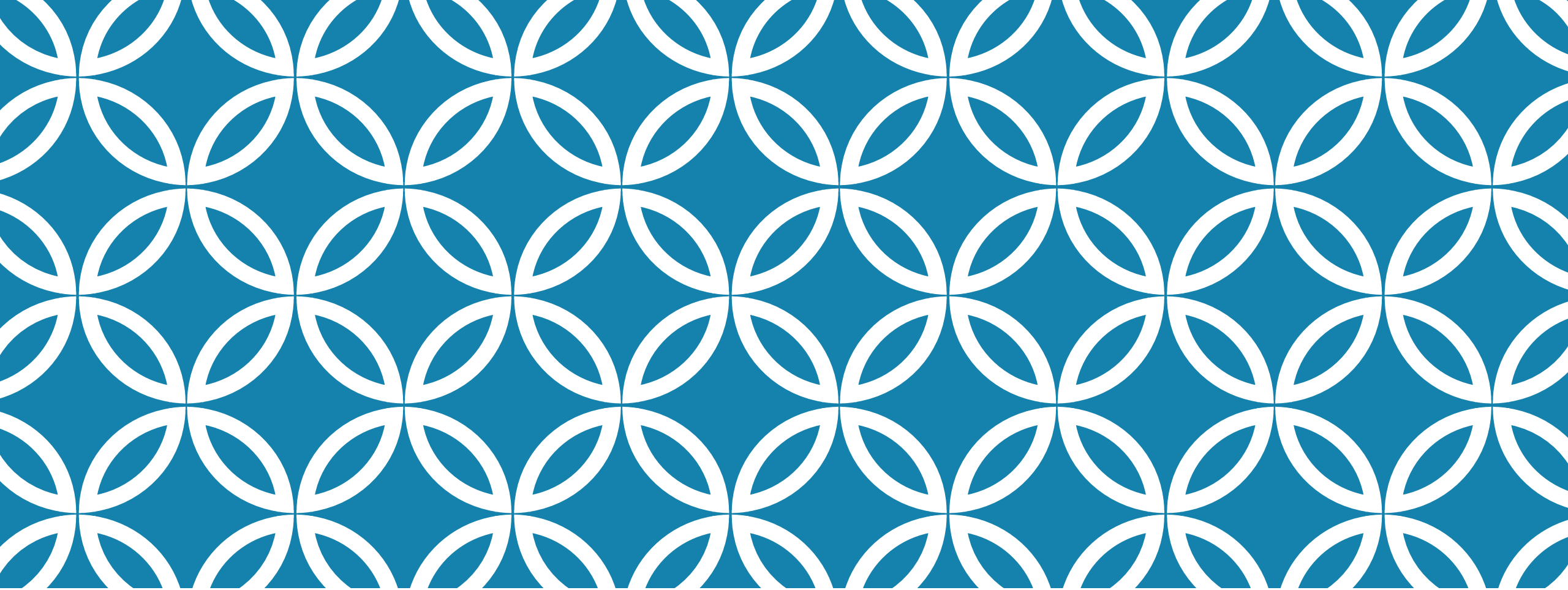
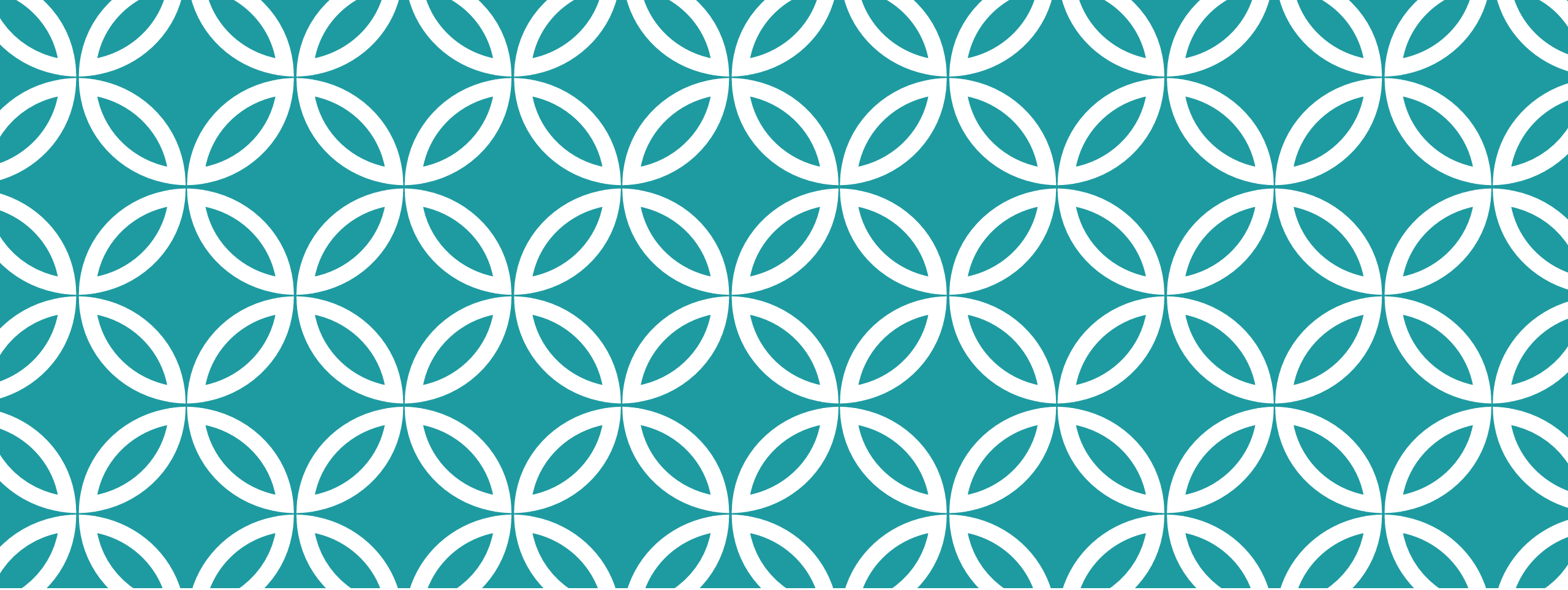


PROGRAMMATION JAVA OBJET

Jérémy PERROUAULT



INTRODUCTION

Les bases de l'objet

INTRODUCTION

Les classes mises à disposition par JAVA se trouvent dans des packages particuliers

- Pour pouvoir manipuler ces classes, il faut **importer** le ou les packages concernés
- D'une manière générale, si la classe n'est pas accessible MAIS qu'elle existe belle et bien
  - Eclipse vous propose de l'importer pour la manipuler

```
import java.util.ArrayList;

import java.util.*;
```

LES CLASSES BASIQUES

Type algorithmique	Type (primitif) JAVA	Classe JAVA
Entier très Court	byte	(java.lang.)Byte
Entier Court	short	(java.lang.)Short
Entier	int	(java.lang.)Integer
Entier Long	long	(java.lang.)Long
Réel	float	(java.lang.)Float
Réel Long	double	(java.lang.)Double
Caractère	char	(java.lang.)Character
Booléen	boolean	(java.lang.)Boolean
Chaine		(java.lang.)String

QUELQUES CLASSES

Classe JAVA	Description
(java.lang.)System.out	Sortie du système (console)
(java.lang.)System.in	Entrée du système (clavier)
java.util.Scanner	Scanner un flux (clavier par exemple)
java.util.ArrayList	Collection d'objets sous forme de tableau dynamique
java.util.LinkedList	Collection d'objets sous forme de liste chainée
java.util.HashMap	Liste de clés-valeurs (ou tableau associatif)
java.util.HashSet	Liste de valeurs sans doublon
java.util.Collections	[Statiques] Fonctionnalités liées aux collections

LES COLLECTIONS

On distingue plusieurs catégories de Collection

- List
  - ArrayList
  - LinkedList
- Map
  - HashMap
  - LinkedHashMap
- Set
  - HashSet
  - LinkedHashSet

LA LISTE CHAINÉE (LINKED)

Très rapide pour insérer et supprimer des données

- Il n’y a pas de tableau
- Pas besoin de créer un décalage

Plus lente pour une lecture

- Il n’y a pas de tableau
- Parcourir chaque élément jusqu’à trouver le bon élément

LA LISTE CHAINÉE (LINKED)

LE TABLEAU DYNAMIQUE (ARRAY)

Très rapide à lire et à parcourir

- C’est un tableau, accès direct via l’indice

A éviter si la liste est trop variable

- Beaucoup d’insertion / de suppression à faire dans un laps de temps court
- L’allocation d’un nouveau tableau est faite à chaque ajout / suppression

LE TABLEAU ASSOCIATIF (MAP)

Fonctionne avec un couple clé-valeur

- Comme si l'indice d'un tableau devenait par exemple une chaine de caractère
  - monTab["jeremy"] au lieu de monTab[5]
- Une seule clé unique
  - Mais une valeur peut être affectée à plusieurs clés différentes

Bien plus gourmand en mémoire

- Il faut stocker la clé en plus de la valeur

Plus lent pour une lecture

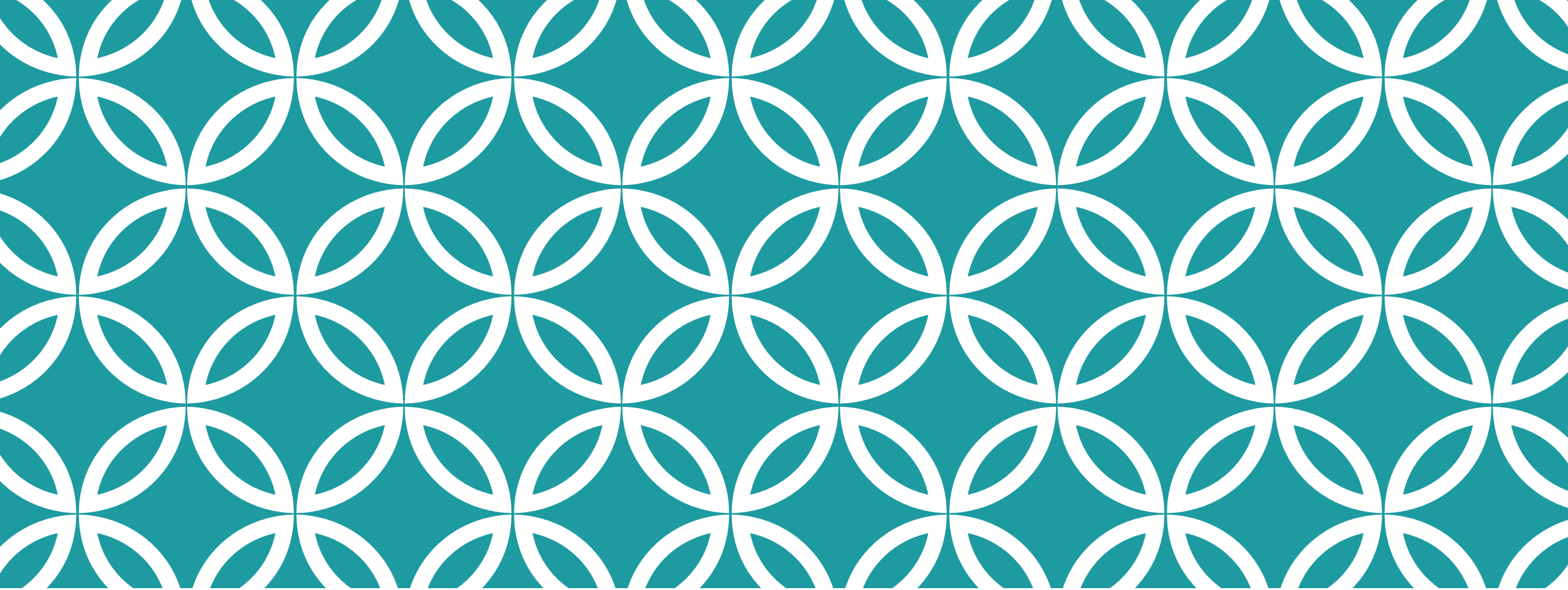
- Il faudra parcourir toutes les données jusqu'à trouver la clé recherchée

LE TABLEAU SANS DOUBLON (SET)

Derrière le tableau dynamique il y a un tableau

Derrière le tableau sans doublon il y a un tableau associatif

Pas d’accès direct à une valeur



ECRIRE UN PROGRAMME

En JAVA Objet

ECRIRE UN PROGRAMME

Les sous-programmes s'appellent désormais des méthodes

Chaque méthode doit avoir une portée définie

- public – accessible par tous
- private – accessible uniquement par la classe l'ayant définie (sa propre méthode)
- protected – accessible par la classe l'ayant définie, et ses enfants (principe d'héritage)

```
public void maMethode() {
}

private void maMethodeInterneQueYaQueMoiQuiYaAcces() {
}
```

- Si la portée n'est pas précisée, c'est une portée « package » qui est présente par défaut

ECRIRE UN PROGRAMME

En algorithmie, le mot-clé « moi » désigne l'instance en cours de manipulation

- En JAVA, ce mot-clé est « this »
- « this » donne accès à tous les attributs et à toutes les méthodes de l'instance

ECRIRE UN PROGRAMME

Déclaration d'une classe (un fichier par classe)

```
public class Personne
{
}
```

Déclaration des attributs

- /!\ TOUT ATTRIBUT DOIT ETRE PRIVÉ OU PROTÉGÉ /!\

```
public class Personne {
  private String nom;
  private String prenom;
  private int age;
}
```

ECRIRE UN PROGRAMME

Déclaration des accès aux attributs

- En lecture et/ou en écriture
- Getters / Setters
- Principe d'encapsulation

```java
public class Personne {
  private String nom;
  private String prenom;
  private int age;

  public String getNom() {
    return this.nom;
  }
  public void setNom(String nom) {
    this.nom = nom;
  }

  public String getPrenom() {
    return this.prenom;
  }
  public void setPrenom(String prenom) {
    this.prenom = prenom;
  }

  public int getAge() {
    return this.age;
  }
  public void setAge(int age) {
    this.age = age;
  }
}
```

ECRIRE UN PROGRAMME

Déclarer une liste de Personne

```
ArrayList<Personne> mesPersonnes = new ArrayList<Personne>();
```

```
Personne maPersonne = new Personne();

maPersonne.setPrenom("Jérémy");
maPersonne.setAge(29);

mesPersonnes.add(maPersonne);
```

ECRIRE UN PROGRAMME

Parcourir une liste (Boucle for)

```
for (int i = 0; i < mesPersonnes.size(); i++) {
  System.out.println(mesPersonnes.get(i).getPrenom());
}
```

Parcourir une liste (Boucle foreach)

```
for (Personne p : mesPersonnes) {
  System.out.println(p.getNom());
}
```

EXERCICE

Implémenter l'algorithme Objet

Créer et implémenter un algorithme de distribution des cartes

- Distribution J1 EQ1 -> J1 EQ2 -> J2 EQ1 -> J2 EQ2
- Distribution des cartes par 3, puis par 2
- Utiliser « java.util.Collections.*shuffle(arrayList)* » pour mélanger les cartes
- Tous les joueurs doivent se retrouver avec 5 cartes
  - Afficher la liste des cartes pour chaque joueur