

L3 SRI - COO



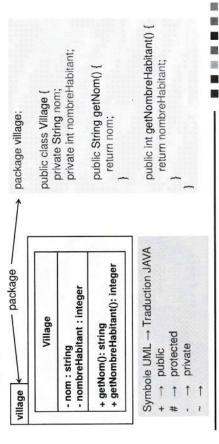
UML / JAVA Cours 1:

diagramme de classe, diagramme de séquence Coder en Java des spécifications UML Les interfaces

Auteur : CHAUDET Christelle

Parallèle UML/Java

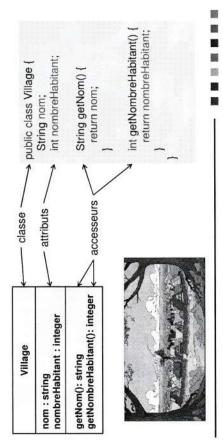
Visibilité = ce qu'un objet a le droit de voir et de modifier. Les droits d'accès de chaque composant sont définis par un mot-clé, dit modifieur.



UML: Diagramme de classes / UML: Diagramme de séquence

Parallèle Diagramme de Classe / Java

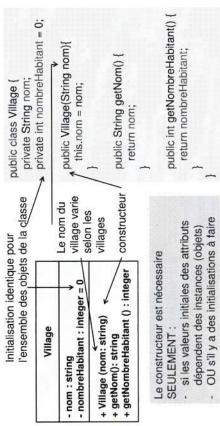
■ Classe = abstraction d'objets ayant des propriétés (état, comportement et relations) communes



UML: Diagramme de classes / UML: Diagramme de séquence

Parallèle UML/Java

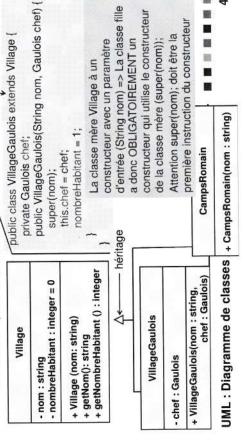
■ Constructeur = permet de créer une instance d'une classe (un objet).



UML: Diagramme de classes / UML: Diagramme de séquence

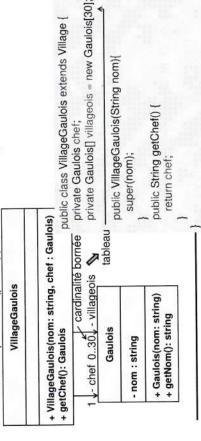
Parallèle UML/Java

Héritage = il s'agit ici de permettre a une classe (dite "classe fille")
de récupérer des attributs, des constructeurs et des méthodes d'une
autre classe (dite "classe mère")



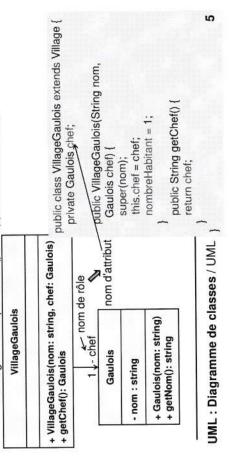
Parallèle UML/Java

 Association = il y a association (dirigée ou pas) entre deux classes, lorsqu'une des deux classes sert de type d'attribut à l'autre, et donc qu'apparaît dans le code de cette dernière l'envoi d'un message vers la première (par l'appel d'une méthode).



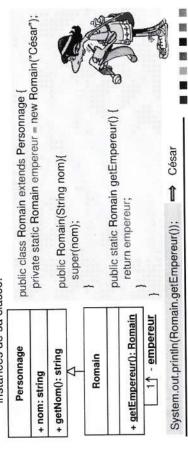
Parallèle UML/Java

 Association = il y a association (dirigée ou non) entre deux classes, lorsqu'une des deux classes sert de type d'attribut à l'autre, et donc qu'apparaît dans le code de cette dernière l'envoi d'un message vers la première (par l'appel d'une méthode).



Parallèle UML/Java

Static = Pour une méthode, le modifieur static indique qu'elle peut être appelée sans instancier sa classe (NomClasse.methode()). Pour un attribut, le modifieur static indique qu'il s'agit d'un attribut de classe, et que sa valeur est donc partagée entre les différentes instances de sa classe.



UML: Diagramme de classes / UML: Diagramme de séquence

UML: Diagramme de classes / UML: Diagramme de séquence

Parallèle UML/Java

membre ou énumérateurs du type énuméré. Existe dans Java Enuméré: type de données qui désigne un ensemble fini de valeurs ordonnées. Ces valeurs sont appelées : éléments, à partir de la version 1.5.

<<enum>> GradeRomain LEGIONNAIRE CENTURION PREFET

public enum GradeRomain { PREFET, LEGAT, CENTURION, LEGIONNAIRE;

System.out.println(GradeRomain.PREFET); Exemple d'utilisation d'un énuméré : public static void main(String[] args) {

PREFET Affichage console

UML: Diagramme de classes / UML: Diagramme de séquence

Parallèle UML/Java

- Par convention que ce soit en UML ou en JAVA
- Les noms de constantes sont entièrement en majuscule, si le nom est composé de plusieurs mots ils seront accolés par le tiret du 8.
 - Ex: COEFFICIENT FORCE
- Majuscule, si le nom est composé de plusieurs mots ils Les noms des types énuméré commencent par une seront accolés en mettant une majuscule à chaque nouveau mot.
 - Ex: GradeRomain
- Les noms des énumérateurs d'un type énuméré sont entièrement en majuscule, si le nom est composé de plusieurs mots ils seront accolés par le tiret du 8. Ex:PREFET

UML: Diagramme de classes / UML: Diagramme de séquence

Parallèle UML/Java

- Par convention que ce soit en UML ou en JAVA
- Les noms de paquetage sont entièrement en minuscule,
- Les noms de classes commencent par une Majuscule, si le nom est composé de plusieurs mots ils seront accolés en mettant une majuscule à chaque nouveau mot. Ex: Village Gaulois
- minuscule, si le nom est composé de plusieurs mots ils variable locale à une méthode) commencent par une Les noms des attributs et des variables (attribut ou seront accolés en mettant une majuscule à chaque nouveau mot.

Ex: nombreHabitant, getNombreHabitant

UML: Diagramme de classes / UML: Diagramme de séquence

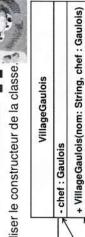
Les objets

+ VillageGaulois(nom: String, chef: Gaulois) + getChef(): Gaulois new VillageGaulois("village des irréductibles Gaulois", nom = "village des irréductibles Gaulois" → villageGaulois:VillageGaulois Pour créer un objet il faut utiliser le constructeur de la classe. VillageGaulois new Gaulois ("Abraracourcix")); chef = abraracourcix nombreHabitant = 1 - chef : Gaulois nom de l'objet type de l'objet package Armoricae; + getNom(): string + getNombreHabitant (): integer VillageGaulois villageGaulois nom : string
 nombreHabitant: integer = 0 public class Armovicae + Village (nom: string)

UML : Diagramme de classes / UML : Diagramme de séquence

Pour créer un objet il faut utiliser le constructeur de la classe 📑





+ getChef(): Gaulois

+ getNom(): string + getNombreHabitant (): integer

+ Village (nom: string)

- nombreHabitant: integer = 0

nom : string

nom = "village des irréductibles Gaulois" villageGaulois:VillageGaulois

chef = abraracourcix nombreHabitant = 1

new VillageGaulois("village des irréductibles Gaulois", nom de la classe ET du constructeur VillageGaulois villageGaulois =

public class Armoricae

package Armoricae;

new Gaulois("Abraracourcix");

CampsRomain[] campsRomain = new CampsRomain[4]; int nombreCampsRomain = 0; UML: Diagramme de classes / UML: Diagramme de séquence

12

Les objets

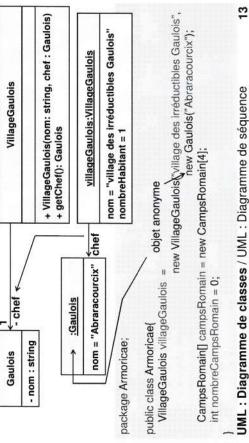
Pour créer un objet il faut utiliser le constructeur de la classe.

public static void main(String[] args) { new VillageGaulois("village des irréductibles Gaulois"); VillageGaulois("village des irréductibles Gaulois") import gaule. Armoricae; → new Armoricae(); public class Test { package test; CampRomain[] campsRomains = new CampRomain[4]; « create » objet anonyme VillageGaulois villageGaulois = int nombreCampsRomains = 0; public class Armoricae(package gaule; :Armoricae

UML: Diagramme de classes / UML: Diagramme de séquence

Les objets

Pour créer un objet il faut utiliser le constructeur de la classe.



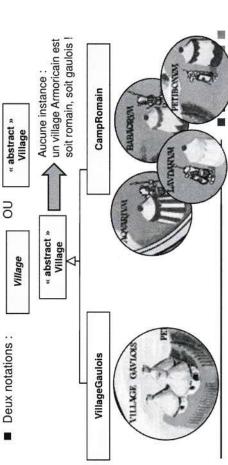
UML: Diagramme de classes / UML: Diagramme de séquence

Les classes abstraites

Classe abstraite : c'est une classe qui n'est pas instanciable.

Elle sert de base à d'autres classes dérivées (héritées)

0



UML: Diagramme de classes / UML: Diagramme de séquence

Les classes abstraites

Classe abstraite : c'est une classe qui n'est pas instanciable. Elle sert de base à d'autres classes dérivées (héritées)



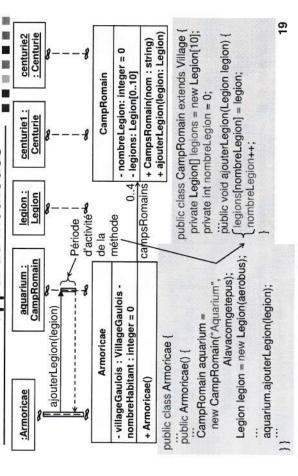
protected int nombreHabitant = 0; public Village(String nom) { public abstract class Village return nombreHabitant UML: Dia + CampsRomain(nom: string) |: Diagramme de séquence public String getNom() { protected String nom; this.nom = nom; return nom; CampsRomain + getNom(): string + getNombreHabitant (): integer chef: Gaulois) # nombreHabitant : integer = 0 + VillageGaulois(nom : string, + Village (nom: string) VillageGaulois # nom : string · chef : Gaulois

public int getNombreHabitant() {

La période d'activité débute quand la méthode est appelée et se termine quand la méthode retourne son paramètre de sortie.

UML: Diagramme de classes / UML: Diagramme de séquence

Interactions entre objets: appel de méthodés



nteractions entre objets: appel de méthodes

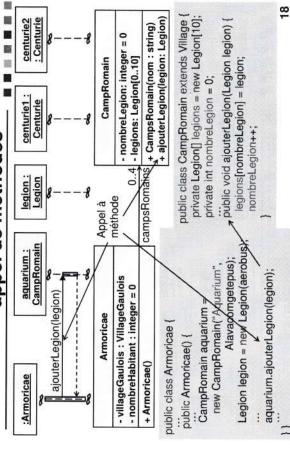


Diagramme de séquence détaillé

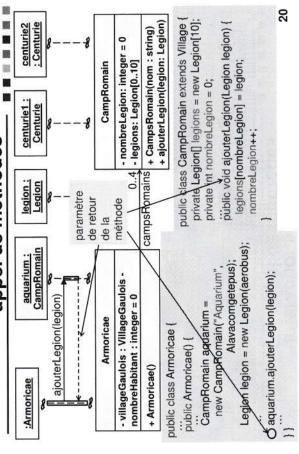
Permet de décrire le comportement interne du logiciel.

Chacun des:

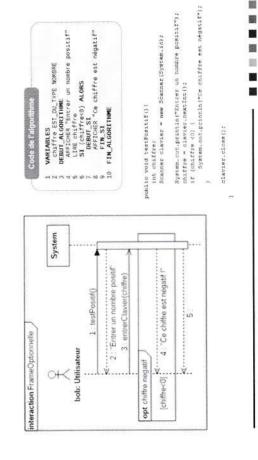
 messages seront spécifiés comme des appels à méthode de la classe réceptrice.

la variable qui sera affectée par la valeur retournée par la méthodes. Le nom de la variable de retour est le nom de retours seront spécifiés comme les retours des

Interactions entre objets : appel de méthodes



Frame Optionnelle



UML: Diagramme de classes / UML: Diagramme de séquence

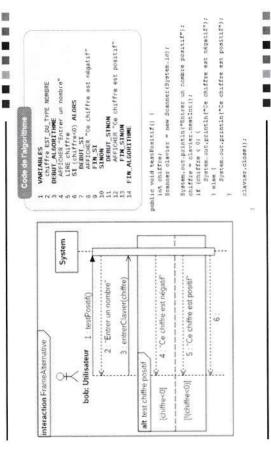
22

Diagramme de séquence Système

- Montre les interactions entre un (ou plusieurs) acteur(s) et le système.
- Si on pense une interaction:
- type « clavier » / « écran » on va utiliser comme nom de message « entrerClavier(valeur) »,
- type interface graphique on va plutôt utiliser des noms de message comme « clicBouton(valeur) », « entrerTexte(valeur) », « selectionner(valeur) » ...
- Permet de faire le prototypage du logiciel.

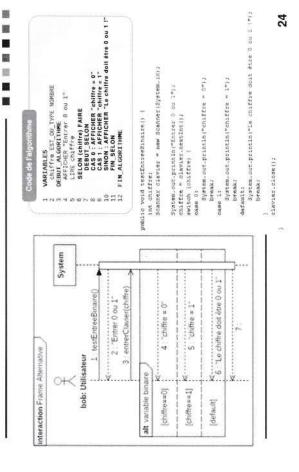


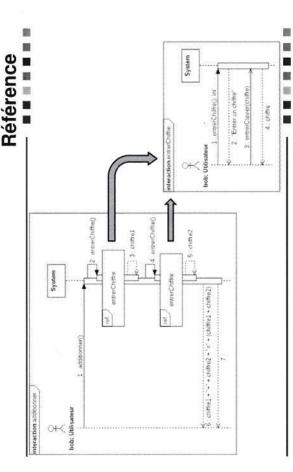
Frame Alternative



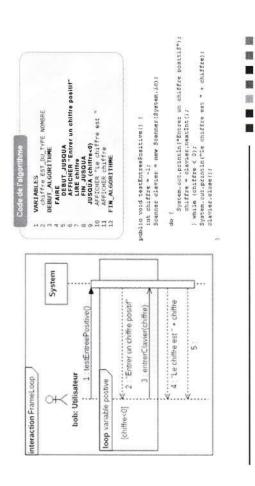
UML : Diagramme de classes / UML : Diagramme de séquence

Frame Alternative





Frame Loop



UML : Diagramme de classes / UML : Diagramme de séquence

for (int i = 0; i < nombreCampsRomains; i++) {
 chaine += campsRomains[i] + "\n\n"; Soldat "Camps romains en Armoricae"; public String afficherCampsRomains(){ :Centurie Interactions entre objets : appel de méthodes public class Armoricae { String chaine = return chaine; :Legion [nombreCampsRomains] toString() :CampRomain lafficherCampsRomains() chaine :Armoricae dool :Utilisateur chaine UML : Diagramme de classes / UML : Diagramme de séquence

UML : Diagramme de classes / UML : Diagramme de séquence

