Séance 1

Surcharge : même nom de fonction différents paramètre

Polymorphisme : une méthode a différents comportements selon les cas

Extends : "est une"

Attribut : l'autre cas, reutilisable dans d'autres classes

Ctrl + maj + O → met à jour les librairies importées

ArrayList : le plus simple d'utilisation

ArrayList<type> nom
(sans espace)

Constructeur → super ⇔ appel le constructeur parent

Class → majuscule au début ⇔ convention

Projet : domotique

↳ Main : main → execution

↳ House : Building

House → extends car type batiment

Room

Component

Séance 2

Dans une pièce mettre pièce
entrée / sortie
bouger

Getter → permet de récupérer les infos dans d'autres classes

Ajout dans Array liste new class dedans

for (int i = 0 ; i < ; i++)
↳ i utilisable que dans la boucle

Il y a des constructeurs auto

Prochaine fois: faire énumération porte fenêtre
placement

Séance 3

Énumération dans Component
car peut être partout

Ajout fridge ⇒ ajout main
⇒ type string
m/ dans

Fridge Récrite de Component
add Component ⇒ possible pour toutes les filles ⇒ traitement mandon } Polymorphisme

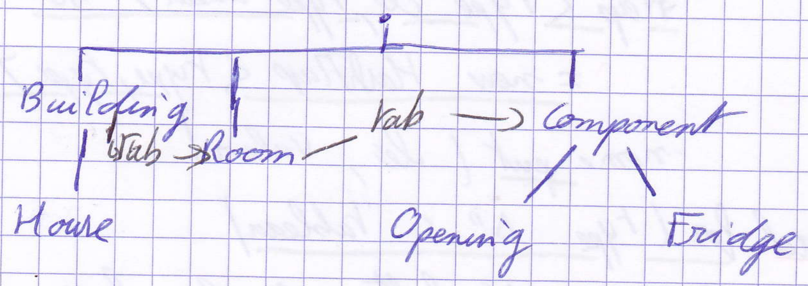
Polymorphisme 1 classe mère et x classes filles
ont celles de la classe mère

Main ⇒ crée maison ⇒ room ⇒ fridge
ajout

Reflexion capacité du code à s'interroger sur lui même
instance of ⇒ regarde si la in class

abstract → methode de la classe abstraite doit être dans toutes les filles

identifica° des pieces (frigo)
↳ enums ou string
↳ modif dans pieces → mettre & chang



Seance 4

this pour parler de la variable de la méthode de l'objet addRoom dans Building car est créé dans building

Hmap → comme les noms mais la clé est un String

Hmap<type de, type valeur> nom = new HashMap<type, type>

nom.put(clé, valeur)

Foreach for(type lp : tableau)

iterateur 1e lettre de la salle

equals verifie aussi le type