ENSAM ÉCOLE NATIONALE SUPÉRIEURE D'ARTS ET MÉTIERS UNIVERSITÉ HASSAN II DE CASABLANCA	Pr S. Sekkate **POO JAVA** TP 2 : Gestion des exceptions	2024-2025

Exercice 1

On souhaite gérer les exceptions afin de s'assurer que la longueur du rayon d'un cercle est bien un réel positif. Dans le cas où une valeur négative est donnée, la longueur du rayon du cercle sera initialisée à son opposé.

Les étapes à suivre sont les suivantes :

1. Écrire la déclaration de la classe **ValRayonValideException** qui permettra d'instancier une exception dès qu'une longueur de rayon négative sera donnée.
2. Écrire la déclaration de la classe **CouleurValideException** qui permettra d'instancier une exception dès que la couleur est un objet nul ou de longueur nulle ou de couleur rouge.
3. Construire la classe **Cercle** en gérant correctement le code de ses méthodes qui sont à risque.
4. Tester la classe Cercle dans la fonction main.

Le squelette de la classe Cercle est donné par :

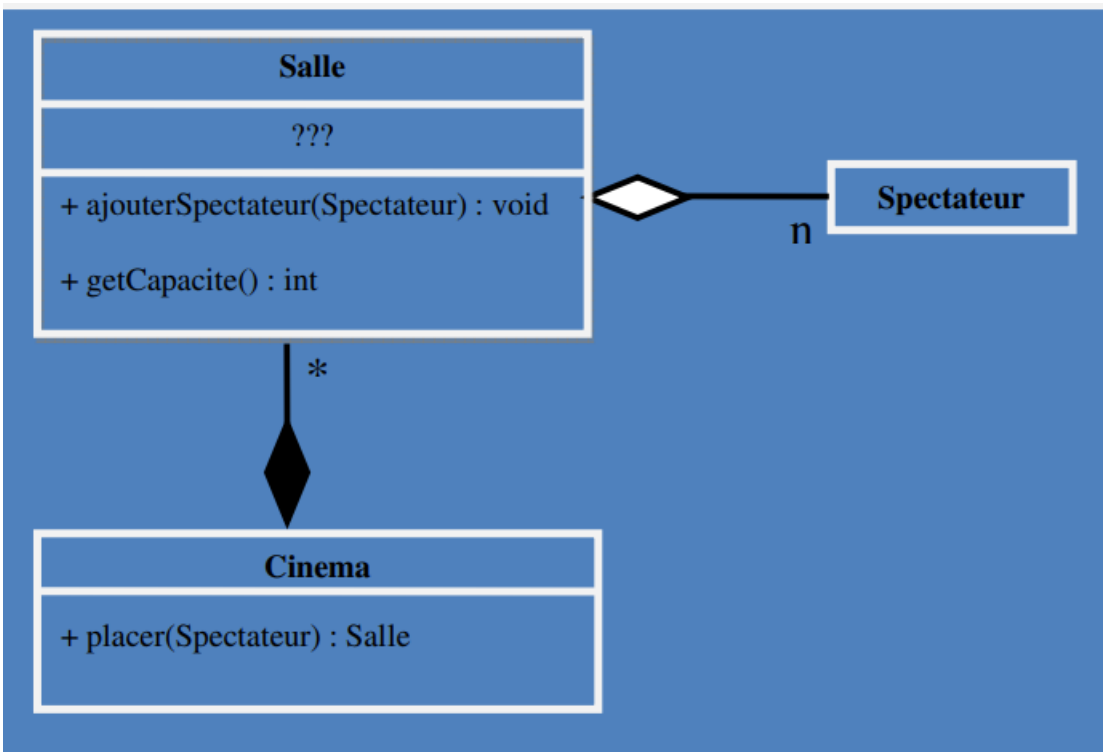
public class Cercle {

private double x, y, rayon;	(x, y) est le centre du cercle dont le rayon est donné par rayon.
private String couleur;	La couleur du cercle
public Cercle(double x, double y, double rayon, String couleur)	Constructeur de la classe
public double getRayon()	
public String getCouleur()	
public void setRayon(double newRayon)	Fonction permettant de modifier la valeur du rayon du cercle. Si la valeur du nouveau rayon donnée par newRayon est négative alors une exception de type ValRayonValideException est levée et traitée.
public void setCouleur(String newCouleur)	Fonction permettant de modifier la couleur du cercle courant, en gérant des exceptions de type NullPointerException si le paramètre newCouleur est nul ou s'il est une chaine de caractères vide ou si newCouleur est de couleur rouge ceci en utilisant la classe CouleurValideException
public String toString()	Fonction permettant d'afficher les caractéristiques du cercle courant

}

Exercice 2

On veut modéliser en Java le fonctionnement simplifié d'un cinéma (cf. figure ci-dessous) qui ne propose qu'un seul film mais dispose de plusieurs salles. Une Salle peut accueillir un nombre fixe (n) de Spectateur. Lorsqu'un spectateur arrive, on le place dans la dernière salle disponible. Si la salle est pleine, alors on ouvre une nouvelle salle. Un Cinema peut ouvrir autant de salles (*) qu'il le souhaite.



1. Définir une classe **SallePleineException** qui sera levée lorsque le cinéma essaye de placer un spectateur dans une salle pleine. Cette exception doit connaître le spectateur qui a déclenché sa levée.
2. Donner l'implémentation de la classe Salle avec ses champs, son constructeur et la méthode **ajouterSpectateur(Spectateur)** qui ajoute un spectateur et lève, sans la traiter, l'exception SallePleineException lorsque la salle est pleine. Il faut choisir la structure de donnée qui vous semble être la plus appropriée (tableau, liste, ...).
3. Donner l'implémentation de la classe **Cinema** avec ses champs, son constructeur et la méthode **placer(Spectateur)** qui ajoute un spectateur à la dernière salle disponible. Si l'exception est levée, alors on créé une nouvelle salle avec sa propre capacité dans laquelle on place le spectateur