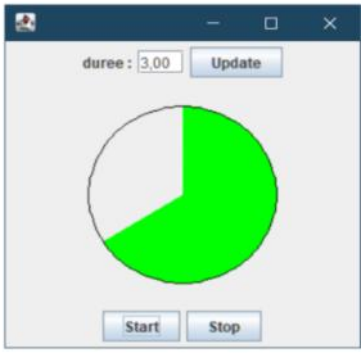
Info402 – Programmation orientée objet II (Java) – 14 mars 2025

Remarques : - Durée 2h00. Feuille A4 recto-verso autorisée. Barème indicatif.
- La clarté de votre copie sera prise en compte.
- Vous devez respecter scrupuleusement la syntaxe et les conventions de Java.
- Il n'est pas nécessaire de signaler les importations des packages (liste des *import* en début de classe).

On veut réaliser une interface graphique pour un minuteur
On suppose que la classe **Minuteur**, décrite en **annexe 3**, existe déjà.



L'interface graphique ressemblera au dessin ci-contre:
- Au centre, occupant la majeure partie de l'interface, on verra, dans un cercle, la proportion de temps restant par rapport à la durée du minuteur.
- Au dessus un champ de saisie et un bouton, permettront de choisir ou modifier la durée du minuteur
- En dessous deux boutons permettront d'arrêter ou de redémarrer le décompte du minuteur

1) (1 pts) Dessiner l'arbre des composants graphiques correspondant à votre interface (l'arbre de composition). La fenêtre principale sera une **JFrame**. Vous devez être consistant avec la question 2.
Vous trouverez dans l'**annexe 1** des précisions sur la taille de la fenêtre et dans l'**annexe 2** une description de la méthode **fillArc** qui permet de dessiner un arc d'ovale.

2) (8 pts) Écrire le code permettant de créer et d'afficher l'interface (sans aucun Listener).Votre code comportera **une méthode main**. Vous trouverez dans l'**annexe 1** des précisions sur la taille.

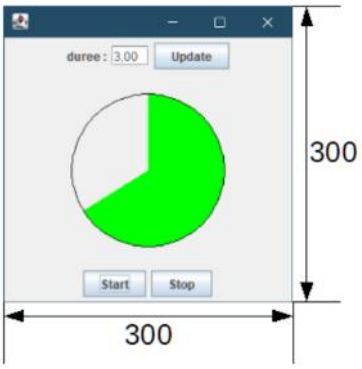
3) (4 pts) L'interface graphique permettra de modifier la durée du minuteur en activant le bouton "**update**".
Écrire le contrôleur (Listener) associé à ce bouton. Ce contrôleur devra récupérer le valeur entrée dans le champ de saisie, afin de modifier la durée du minuteur (en appelant la **méthode reset du minuteur**).
Vous préciserez les modifications à apporter dans le code de la question 2 pour associer ce contrôleur aux composants.

4) (4 pts) Sauvegarde et récupération d'une instance du Minuteur.
On veut pouvoir sauvegarder et restaurer les versions du minuteur les plus fréquemment utilisées.
Ajouter à cet effet à la classe Minuteur :
- une méthode **void sauvegarder(String path)** qui permettra de sauvegarder l'état du minuteur dans le fichier dont le chemin est passé en paramètre.
- une méthode **void charger(String path)** qui permettra de récupérer l'état du minuteur précédemment sauvegardé dans le fichier dont le chemin est passé en paramètre.

5) (3 pts) Quand on clique sur le bouton **start**, on va démarrer (ou redémarrer) le minuteur (**appel de la méthode start** sur le minuteur, ce qui fera passer le passer le booléen **running** du minuteur à vrai).
Écrire le contrôleur (Listener) associé au bouton start. Le contrôleur appellera la méthode start sur le minuteur et devra créer, si le minuteur n'est pas déjà démarré, une tâche permettant de rafraîchir périodiquement l'affichage du minuteur (toutes les secondes, par exemple), tant que le minuteur n'est pas arrêté (le booléen **running** du minuteur est a vrai) ou terminé.

Préciser les modifications à apporter au code de la question 2 pour associer ce contrôleur aux composants.

Annexe 1 : taille de la fenêtre



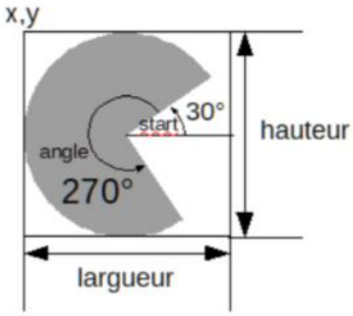
Annexe 2 : méthode **fillArc** de la classe **Graphics**

Pour dessiner un arc d'ovale il est possible d'utiliser, sur les objets de type **Graphics**, la méthode **fillArc** dont le prototype est le suivant :

public void fillArc(int x, int y, int largeur, int hauteur, int start, int angle) où

- **x** et **y** sont les coordonnées du coin supérieur gauche du rectangle englobant
- **largeur** et **hauteur** correspondent à la largeur et la hauteur du rectangle englobant l'ovale
- **start** est l'angle en degré du début de l'arc avec l'horizontale (valeurs positives dans le sens antihoraire et négatives dans le sens horaire)
- **angle** est l'angle en degrés entre le début et la fin de l'arc (valeurs positives dans le sens antihoraire et négatives dans le sens horaire)

Le dessin ci-contre correspondrait à un arc de disque avec un angle (start) de départ de 30° et et un angle de 270° :



Annexe 3 : description de la classe classe **Minuteur**

```
public class Minuteur implements Serializable{
    // duree du minuteur en minutes
    private double minutes;
    // temps restant en minutes
    private double restant;
    // indique si le minuteur est arrete ou non
    private boolean running;

    // constructeur
    public Minuteur() {
        this(3.0);
    }
    public Minuteur(double minutes) {
        this.minutes = minutes;
        this.restant = minutes;
        this.running = false;
    }

    // getters
    public double getMinutes() ;
    public double getRestant() ;
    public boolean isRunning() ;
    // setters
    private void setMinutes(double minutes) ;
    private void setRestant(double restant) ;
    private void setRunning(boolean running) ;

    // demarre ou redemarre le decompte
    public void start() ;
    // arrete le decompte
    public void stop() ;

    // reinitialise le décompte
    public void reset() {
        this.reset(this.minutes);
    }
    // reinitialise le décompte en changeant la duree du minuteur
    public void reset(double minutes) {
        this.setMinutes(minutes);
        this.setRestant(minutes);
    }
}
```