Code branche **INFOR**	**Ministère de l'Éducation nationale, de l'Enfance et de la Jeunesse** EXAMEN DE FIN D'ETUDES SECONDAIRES TECHNIQUES **Régime technique - Session 2015/2016**	
Épreuve écrite	*Branche* **Informatique**	*Division / Section*
Durée épreuve **3 heures**	**JAVA**	**GE**
Date épreuve 16.9.2016		

Dans votre répertoire de travail (à définir par chaque Lycée), vous trouverez un sous-dossier nommé **EXAMEN_GE**. Renommez ce dossier en remplaçant le nom par votre code de l'examen (exemple de notation : **LTXX_GE1_07**). *Tous vos fichiers devront être sauvegardés à l'intérieur de ce sous-dossier, qui sera appelé 'votre dossier' dans la suite !*

Question 1 — 10 points

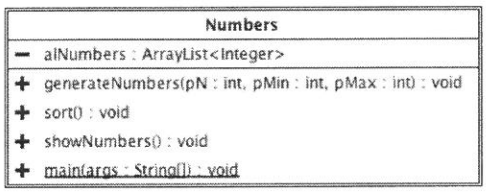
Ouvrez le projet **`Question1`** de votre dossier. Ce projet contient la classes **`Numbers`** qui permet de gérer une liste de nombres.

Consignes et informations générales :

- L'environnement de programmation est à votre choix (Unimozer ou NetBeans).
- Une méthode **`main`** de la classe **`Numbers`** vous permet de tester vos méthodes modifiées.

Travail à réaliser:

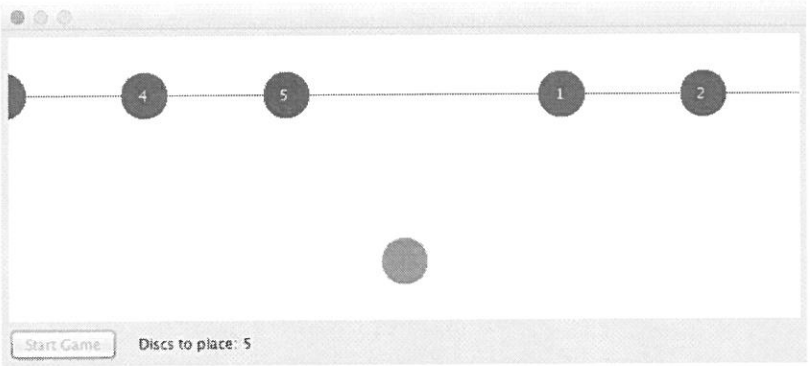
- Développez la méthode **`generate`**() permettant de remplir la liste **`alNumbers`** avec **`pN`** nombres aléatoires compris dans l'intervalle [**`pMin`**; **`pMax`**]. **`pN`**, **`pMin`** et **`pMax`** étant 3 paramètres entiers de la méthode. [2p]
- Développez la méthode **`sort`**() permettant de trier la liste **`alNumbers`** dans l'ordre croissant à l'aide de l'algorithme de tri par sélection directe. [8p]



Question 2 — 50 points

Développez le jeu **"Discs"** dont le but est de placer un nombre de disque donnés dans une suite de disques, sans pour autant toucher les disques déjà placés.

Les disques placés, de couleur bleue, sont numérotés et se déplacent horizontalement vers la droite à vitesse constante. En cliquant sur un bouton quelconque de la souris, le joueur lance le disque rouge. Celui-ci se déplace verticalement vers le haut.



Dans la suite, vous aller découvrir le détail des classes. Vous trouverez dans votre dossier une version exécutable (`Question2Solution.jar`) dans le sous-dossier `Question2` afin de vous familiariser avec le fonctionnement du programme.

Classe Disc — [6p]

La classe `Disc` représente, comme son nom l'indique, un disque de rayon (`radius`) 20 et d'une couleur (`color`). Sa position sur l'interface de dessin est représentée par l'attribut `position`. [1p]

- Le constructeur de la classe prend comme paramètres la position et la couleur du disque. Il initialise les attributs avec les valeurs des paramètres et initialise le rayon du disque à la valeur 20. [1p]
- Les méthodes `getPosition`() et, `getRadius`() sont les accesseurs pour les attributs respectifs. [1p]
- Les méthodes `setX`(), resp. `setY`() permettent de modifier la position du disque suivant l'abscisse `x` resp. l'ordonnée `y` donnée. [1p]
- La méthode `draw`() est responsable de l'affichage d'un disque. Un disque est visualisé par un disque de rayon `radius`, dont le centre se trouve à la position `position`. [1,5p]
- La méthode `move`() est responsable du déplacement d'un disque. Toutefois, cette méthode est vide dans cette classe et sera implémentée dans les sous-classes. Le paramètre `size` représente une taille qui sera expliqué dans les sous-classes. [0,5p]

Classe PlacedDisc — [5p]

La classe `PlacedDisc` est une sous-classe de la classe `Disc`, représentant un disque déjà placé dans la suite de disques.

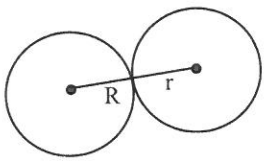
- L'attribut `number` représente le numéro du disque dans la suite de disques, suivant l'ordre dont ils ont été ajoutés à la suite. La méthode `getNumber()` étant l'accesseur de cet attribut. [1p]
- Le constructeur prend comme paramètres la position ainsi que le numéro du disque. Il fait appel au constructeur de sa classe mère pour instancier un disque de couleur bleue, de centre passé en paramètre. [1p]
- La méthode `move`() est responsable du déplacement horizontal du disque. Un disque se déplace de 5 unités vers la droite à chaque appel de cette méthode.
  Dans ce cas ci, le paramètre `size` représente la largeur de la surface de jeu. Si un disque sort de la surface de jeu, alors il est replacé à gauche de celle-ci, son abscisse correspond à la valeur négative du rayon. [1,5p]

- La méthode **draw**() est responsable de la visualisation du disque. Elle fait appel à la méthode **draw**() de sa classe mère et affiche ensuite en couleur blanche le numéro du disque à l'intérieur du disque. Le nombre à afficher est décalé de 5 unités vers la gauche et de 5 unités vers le bas par rapport au centre du disque. [1,5p]

Classe DiscToBePlaced [5p]

La classe **DiscToBePlaced** est une sous-classe de la classe **Disc**. Elle représente un disque que le joueur doit encore placer.

- L'attribut **moving** indique si le disque est en mouvement. La méthode **isMoving**() étant l'accesseur de l'attribut. [1p]
- Le constructeur prend comme paramètre la position initiale du disque. Il fait appel au constructeur de sa classe mère pour instancier un disque de couleur rouge. [0,5p]
- La méthode **launch**() est responsable du lancement du disque et modifie l'attribut **moving** à vrai. [0,5p]
- La méthode **move**() est responsable du déplacement vertical du disque. Si le disque a été lancé, il se déplace de 8 unités vers le haut à chaque appel de cette méthode. [1,5p]
- La méthode **isTouchingDisc**() renvoie vrai si le disque actuel touche une disque **d** passé en paramètre. Deux disques se touchent, lorsque la distance de leurs centres respectifs est inférieure ou égale à la somme de leurs rayons respectifs. [1,5p]

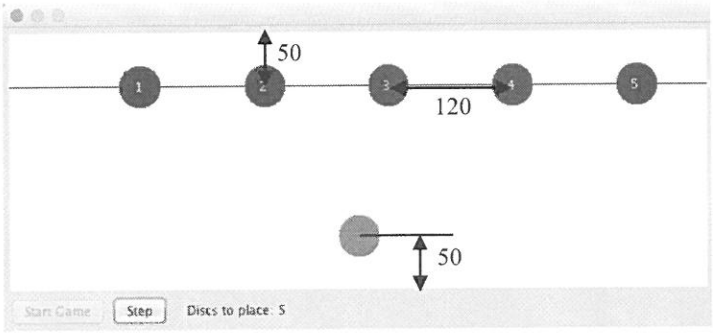


Classe Game [21p]

La classe **Game** est responsable de la gestion du jeu. Cette classe est représentée par: [1p]

- une **ArrayList** contenant les différentes disques placés;
- l'attribut **discToPlace**, représentant le disque que le joueur doit placer;
- l'attribut **gameRunning**, indiquant si le jeu est en cours;
- l'attribut **nbrDiscsToPlace**, indiquant le nombre de disque que le joueur doit placer;
- l'attribut **initialNbrPlacedDiscs**, indiquant le nombre initial de disque déjà placés;
- l'attibut **yPos**, indiquant la hauteur à laquelle les disques placés défilent.

- Le constructeur prend comme paramètres la largeur et la hauteur de la surface de jeu. Il initialise les attributs de la façon suivante : [3,5p]
  - au début du jeu 5 disques sont déjà placés et le joueur doit en placer 5;
  - les disques déjà placés (**PlacedDisc**) se trouvent à 50 unités du bord supérieur de la surface de jeu;
  - le premier disque que le joueur doit placer (**DiscToBePlaced**) est centré horizontalement et se trouve à 50 unités du bord inférieur de la surface de jeu;
  - les disques déjà placés sont instanciés et ajoutés à la liste. La distance entre les centres des disques est de 120 unités.



- Les méthode **isGameRunning**() et **getNbrDiscsToPlace()** sont les accesseurs des attributs respectifs. [1p]
- La méthode **draw**() est responsable de la visualisation du jeu. Elle dessine en premier lieu une ligne droite noire, indiquant le chemin, par lequel les disques placés défilent. La largeur (**pWidth**) passée en paramètre représente la largeur de la surface de jeu.
  Ensuite, si le disque à placer est instancié, alors celui-ci est dessiné.
  Pour enfin dessiner toutes les disques contenues dans la liste. [4p]
- La méthode **moveDiscs**() est responsable du mouvement des disques. Elle prend en paramètre la largeur et la hauteur de la surface de jeu. [10p]
  - Tout d'abord, elle déplace le disque à placer.
  - Ensuite, elle déplace tous les disques déjà placés. En même temps elle contrôle si un disque déjà placé touche le disque à placer. Si tel est le cas, le jeu est arrêté.
  - Dans le cas où le jeu est encore en cours (à contrôler) et que le disque à placer se trouve à la hauteur des disques placés, un nouveau disque de type **PlacedDisc** est créé à l'endroit où le disque à placer se trouve et est ajouté à la liste.
    Le nombre de disques à placer est réduit de 1.
    Si le joueur a placé tous ces disques, alors le jeu est arrêté et l'attribut **discToPlace** est mis à **null**. Si le joueur n'a pas placé tous ces disques, alors un nouveau disque de type **DiscToBePlaced** doit être créé au point de départ des disques à placer.
- La méthode **launch** () permet de lancer un disque à placer s'il a été instancié (à contrôler). [1,5p]

Classe DrawPanel [3p]

La classe **DrawPanel** (de type **JPanel**) est responsable de la visualisation du jeu. Elle contient:

- l'attribut **game** et son manipulateur **setGame**();
- la méthode **paintComponent**() qui affichage d'abord un fond blanc pour ensuite visualiser le jeu.

Classe MainFrame [10p]

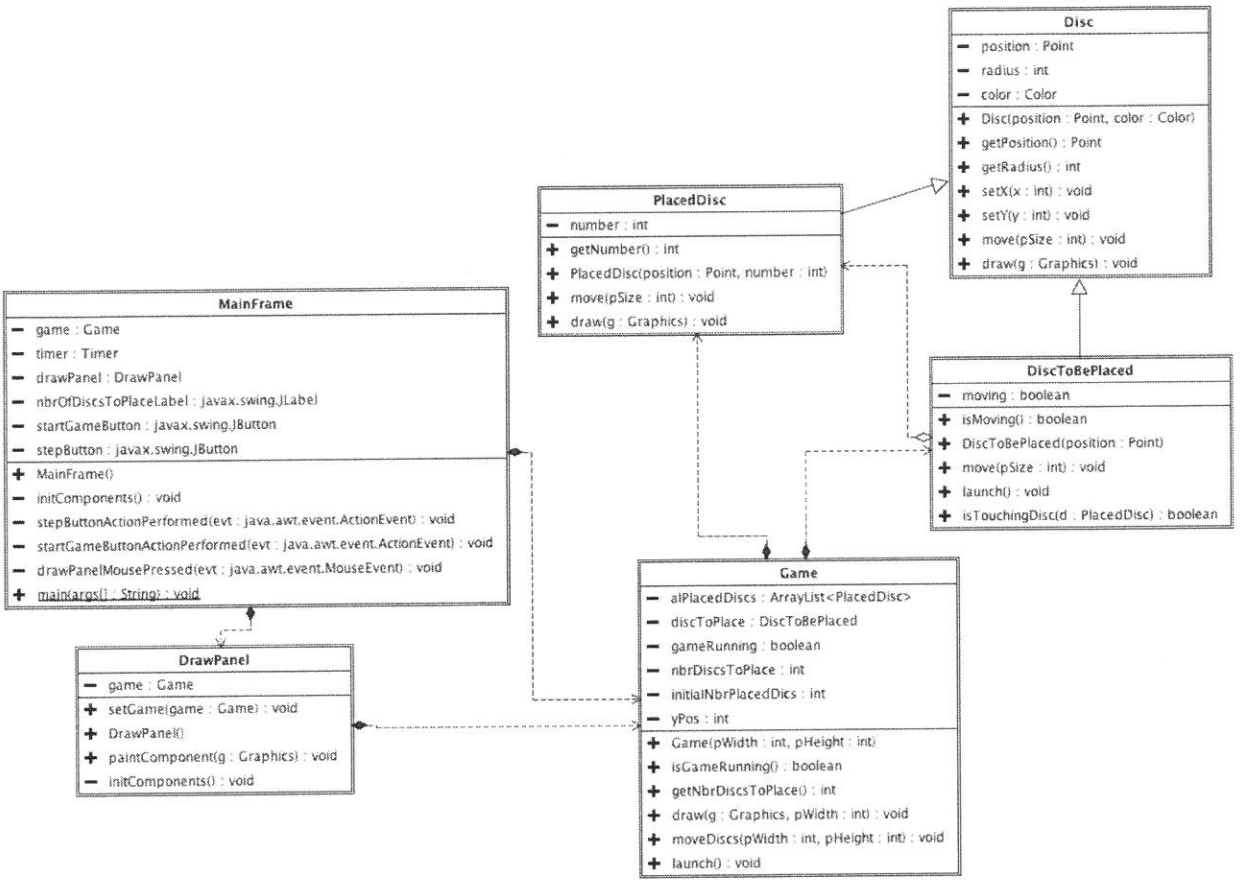
L'interface est à créer suivant la version exécutable fournie et l'image se trouvant au début la question. [1p]

Elle contient les attributs suivants: [1p]

- **game** de Type **Game**, représentant le jeu en lui même;
- **timer** de type **Timer**, représentant le minuteur permettant le mouvement automatique des disques;

- Le constructeur [1p]
  - instancie le minuteur, qui actionne chaque 50ms le bouton **stepButton**;
  - cache le libellé **nbrOfDiscsToBePlacedLabel** et le bouton **stepButton**.
- Lorsque le joueur appuie sur le bouton **startButton**, un nouveau jeu est créé dont la largeur et la hauteur de la surface de dessin **drawPanel** sont passés en paramètre. Le **drawPanel** est informé de l'instance du jeu. Le libellé **nbrOfDiscsToBePlacedLabel** est rendu visible et le nombre de disques à placer restants y est affiché. Le bouton **startButton** est désactivé et le minuteur est activé. [3p]
- Lorsque le joueur clique sur la surface de dessin avec un bouton quelconque de la souris, le disque à placer est lancé. [0,5p]
- Lorsque le bouton **stepButton** est actionné, il faut contrôler si le jeu est en cours. Si tel est le cas, tous les disques sont mis en mouvement, la surface de jeu est redessinée et le libellé avec le nombre de disque à placer restant est mis à jour. Dans le cas contraire, le minuteur est arrêté et le bouton **startButton** est activé. [3,5p]

Diagramme de classes



Enseignement secondaire technique
Division technique générale
Examen 13GE

Liste des composants et classes connus

Liste des composants (propriétés, événements et méthodes) et classes à connaître pour l'épreuve en informatique à l'examen de fin d'études secondaires techniques - division technique générale.

Package	Classe	Details	Remarques / Constantes
javax.swing	JFrame	Méthodes - setTitle(...) / getTitle() NetBeans Object Inspector Property - title	
	JButton JLabel JTextField	Méthodes - setText(...) / getText() - getX() / getY() - getWidth() / getHeight() - setVisible(...) - setEnabled(...) Événement - actionPerformed NetBeans Object Inspector Property - icon	- le libellé *JLabel* peut aussi être utilisé pour visualiser des images via la propriété « icon » de l'inspecteur objet de NetBeans (le composant *JTextField* ne possède pas de propriété « icon »).
	JSlider	Méthodes - setMinimum(...) / getMinimum() - setMaximum(...) / getMaximum() - setValue(...) / getValue() Événement - stateChanged	
	JPanel	Méthodes - setVisible(...) - setEnabled(...) - setBackground(...) / getBackground() - getWidth() / getHeight() - getGraphics() - paintComponent(Graphics g) - repaint() Événements - MousePressed / MouseReleased - MouseDragged / MouseMoved	- *JPanel* est utilisé pour regrouper d'autres composants visuels et pour réaliser des dessins. - Lors de la réalisation de dessins, la méthode **`public void paintComponent (Graphics g)`** est à surcharger.
javax.swing	JList	Méthodes - setListData(...) - getSelectedIndex(...) / setSelectedIndex() Événement - valueChanged NetBeans Object Inspector Properties - model - selectionMode (SINGLE)	- *JList* est utilisé surtout pour afficher le contenu d'une liste *ArrayList*.
java.awt.event	ActionEvent	- Ce type d'objet est uniquement utilisé dans les méthodes de réaction ajoutées de manière automatique à l'aide de NetBeans.	
	MouseEvent	Méthodes - getX() / getY() - getPoint() - getButton()	Constantes - BUTTON1 - BUTTON2 - BUTTON3

Package	Classe	Details	Remarques
javax.swing	Timer	Constructeur - Timer(int,ActionListener) Méthodes - start() - stop() - setDelay(...) - isRunning()	
		- Comme *ActionListener* il faut utiliser celui d'un bouton. Exemple : `Timer timer = new Timer(1000,stepButton.getActionListeners()[0]);`	
java.awt	Graphics	Méthodes - drawLine(...) - drawOval(...) / fillOval(...) - drawRect(...) / fillRect(...) - drawString(...) - setColor(...) / getColor()	
	Color	Constructeurs - Color(...)	
	Point	Constructeurs - Point(...) Attributs (publics) - x et y	
java.util	ArrayList	Méthodes - add(...) - clear() - contains(...) - get(...) - indexOf(...) - remove(...) - set(...) - size() - isEmpty() - toArray()	- `Object[] toArray()` est employé uniquement pour passer les contenus d'une liste à la méthode *setListData(...)* d'une *JList*.
java.lang	String	Méthodes - equals(...) / compareTo(...) - indexOf(...) - valueOf(...)	
	Integer Double	Méthodes - equals(...) / compareTo(...) - valueOf(...)	
	Math	Méthodes - abs(...) - round(...) - random() - sqrt(...) - pow(...) - sin(...), cos(...), tan(...)	Constante: - PI
	System	Méthode - out.print() - out.println()	