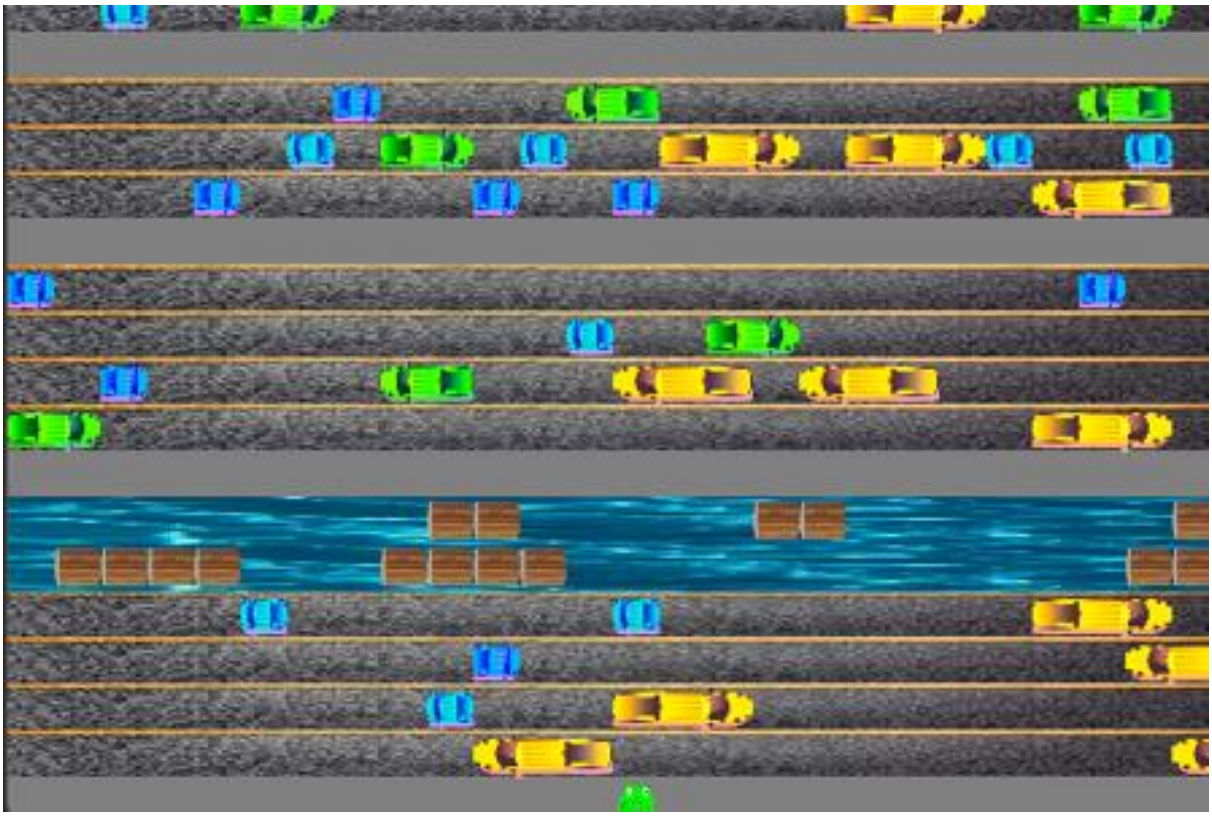
# Projet java -FROGGER-

Slimane mesbah

Jules Rohault de Fleury



Professeur :  Alice JACQUOT & Huyen NGUYEN

1. Frog

```
// 1. move the Frog :
switch(key) {
    case up:
        this.caseFrog = new Case(this.caseFrog.absc,this.caseFrog.ord+1); // or + case.size
        break;
    case down:
        if (this.caseFrog.ord-1 >= this.game.getMinLine())
            this.caseFrog = new Case(this.caseFrog.absc,this.caseFrog.ord-1);
        break;
    case left:
        if (this.caseFrog.absc-1>=0)
            this.caseFrog = new Case(this.caseFrog.absc-1,this.caseFrog.ord);
        break;
    case right:
        if (this.caseFrog.absc+1<this.game.width)
            this.caseFrog = new Case(this.caseFrog.absc+1,this.caseFrog.ord);
        break;
}
```

Dans cette partie : pour pouvoir bouger la grenouille, on a créer une nouvelle case qui aura la position précédante + ou – ( la direction qu'on a eu en cliquant sur les Button flèches du clavier )

Et enfin on mis a jour les  attribut et le jeu

2. Timer :

```
// timer update():
if(this.isGameOn){
    this.graphic.setTimerText("time:"+(float)Math.round(this.timer*10)/10+"s");
    this.graphic.displayTimer();
}
```

Afin d'afficher les temps en second, et pas en M second, par la fonction update qui est appelé toute les tick d'horloge, on converti ce temp, en second et enfin pour avoir un affichage clair, pour l'utilisateur, ce timer est rondi afin d'avoir un entier .

3. ImageG :

```java
public class ImageG {
    public BufferedImage image;


    public ImageG( String imageName){
        try {

            this.image = ImageIO.read(ImageG.class.getResource("/images/" + imageName));
        }
        catch(IOException exc) {
            exc.printStackTrace();
            this.image=null;
        }
    }


}
```

La classe image contient un attribut du type **`BufferedImage`** et un seul constructeur qui prend en parametre le nom de l'image qui le converti ensuit en « chemin vers ce fichier image », un des problème qu'on rencontre est de trouver le chemin vers ce fichier, car le dossier courent est le dossier out pour les fichier exécutable, et pour cela on a proposé deux solultion, une est de mettre les image dans le même package que la class **`ImageG`**, mais cette dernière, rend le code moins organisé. Et par conséquent, on a utiliser le chemin vers le projet lui-même, et puis, vers le package **`src/images`** . avec la methode **`class.getResource`** Package et finalement entoure par un **`try-catch`** exige par le methode **`Image.readIO`**

1. Class Element :

```
//public ArrayList<ImageG> image;
public ArrayList<String> props;
```

La l'utilisation d'un attribut image pour tout les éléments, va pousser le code a créer des objet de class Image, et cela rend le code moins fluid, voir inutilisable après avoir créer autant de voiture que le programme peut gérer rapidement, c'est pour cela que l'objet image est créer une fois est utiliser pour différent élément, par l'intermédiaire **`props`** une chaine de caractère représentant le nom de l'objet image

4. Class voiture :

```
public class Car {
    protected Game game;
    protected Case leftPosition;
    protected boolean leftToRight;
    protected int length;
    protected Color color = Color.pink;//new Color(1
    protected int carType; //1 for regular car, 2 fo
    protected ArrayList<String> props;
    protected boolean hasBackGround;
    protected boolean frogOnIt=false;
```

Pareil pour élément, **`l'Arraylist images`** est changer a une liste de String afin d'eviter la duplication des image

```
public void setImage() {
     if (this.carType == 1) {
        for ( int i=0; i<this.length;i++) {
            if (leftToRight) {
                this.image.add(new ImageG("c"+this.length+"l"+i+".png"));
                this.props.add("c"+this.length+"l"+i);

            } else {
                this.image.add(new ImageG("c"+this.length+"r"+i+".png"));
                this.props.add("c"+this.length+"r"+i);
            }
        }
    } else { //Considering there's only two car types, no need to check if it's ==
    for ( int i=0; i<this.length;i++) {
            this.image.add(new ImageG("rondin.png"));
            this.props.add("rondin");
        }
    }
}
```

La génération du nom de l'image est généré en fonction de la **`direction`** de la voiture ainsi que sa taille, finalement en distinguant les voiture **`car`** et les **`rondin`**

Par le code suivant :

```
//Set the image:
switch(this.laneType) {
    case 1: this.image = new ImageG("road.jpg"); break;
    case 2: this.image = new ImageG("water.jpg"); break;
    default: this.image = new ImageG(""); break;
}
```

### 4.1. **`Car.move()`**

```
public void move() {

    //Check if car is on it
    this.frogOnIt = (this.carType == 1) && this.occupyCase(this.game.getFrogCase());

    this.leftPosition = new Case(this.leftPosition.absc + (leftToRight ? 1 : -1), this.leftPosition.ord);

    //Update graphics
    this.addToGraphics();
}

/**
 * Pause car
```

Si la voiture est du type ( **`rondin`**) lors de son mouvement, la **`frog`** doit bouger dans la même **`direction`** et distance que les rondin bouge, et cela sauf si la **`Frog`** est sur le **`rondin`**

5. **Infinity** :

```
//If lowestLine is too far below, remove lines
while(this.lowestLine < curHeight-this.linesBelow) {
    this.lanes.remove(0);
    this.lowestLine++;
}
```

Les développements suivants se sont concentrés sur les ajouts des parties 3 et 4. Le défilement infini nécessita de revoir la manière dont les lignes **`Lanes`** étaient stockées dans **`l'environnement`**, avec l'ajout et retrait régulier des lignes d'après le scrolling (et le retrait de la condition de victoire). Pour éviter la gestion de toutes les lignes derrière, une limite à 5 lignes de recul existe (au-delà, les lignes antérieures ne sont plus chargées et la **`frog`** ne peut pas reculer plus)

6. **Game levels** :

```
int minSpeedInTimerLoops = 1;
double defaultDensity = 0.1;
```

Pour avoir des niveau différent pour le jeu, les attribut **`minSpeed`** et **`defaultDensity`** seront incrémente pour chaque 10 ligne suivante, jusqu'à ce que le niveau autant de voiture que la **`Frog`** peut bouger difficilement, dans ce cas la, le niveau du jeu est en fonction du temps et score