Le monde des Bugs

Le monde des Buggles a été inventé et proposé par Franklyn Turbak, du Wellesley College. Les Bugs (punaises) en sont les héritières belfortaines. Ce sont de petites bêtes craintives qui comprennent des ordres simples, et offrent des possibilités d'interaction avec le monde : se déplacer, colorier le sol, se cogner à des murs, etc.

Dans chacun des exercices qui vous seront proposés, vous devrez donner des ordres à vos bugs pour faire en sorte que le monde ressemble à l'objectif fixé. Par exemple dans le premier exercice, vous devrez instruire votre bug pour qu'il avance simplement de quelques pas et ramasse un biscuit.

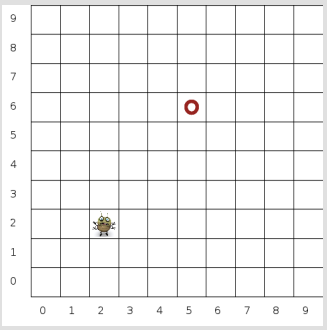
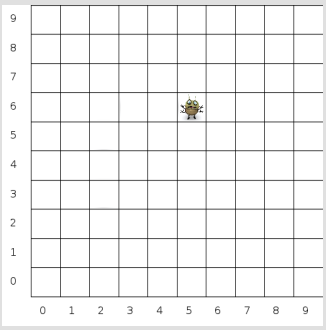
En plus de la description textuelle, l'objectif à atteindre vous sera toujours présenté sous forme d'image, de sorte à pouvoir plus facilement comparer votre résultat à celui attendu.

Une description détaillée des possibilités d'action des Bugs et des méthodes (instructions) correspondantes est donnée au bas de cette page.

Premier défi : nourir un Bug

Les objectifs de ce premier exercice sont :

- Se familiariser avec le monde des Bugs
- Écrire un premier programme simple de contrôle du Bug.

État initial	**Résultat attendu**
• Un Bug dans un monde de 10x10 cases. • Un biscuit sur une case différente de celle du Bug.	• Le Bug s'est déplacé sur la case du biscuit. • Le bug a ramassé le biscuit.
	

À faire

- **Récupérer** :download:`l'archive du programme <code/TP_Bugworld.tar.gz>` et **l'enregistrer** dans le dossier du TP.
- **Se placer** dans le dossier puis **décompresser** l'archive :

```
tar zxvf TP_Bugworld.tar.gz
```

- Le programme comprend plusieurs classes et fichiers
  - Une archive **BugWorld.jar** contenant les classes *BuggleWorld* et *Buggle*.

- Un fichier **Main.java** contenant la définition de la classe principale, responsable de l'initialisation et du lancement du programme.
- Un fichier **Bug.java** contenant la définition de la classe *Bug*. La classe *Bug* hérite ses caractéristiques de *Buggle*, mais possède une méthode qui lui est propre : **enRoute()**. C'est dans cette méthode que vous écrirez votre programme de contrôle de(s) Bugs.
- Un fichier **mazeData.txt** contenant la position des *murs* et des *biscuits*.

Compiler le programme :

```
javac -cp .:BugWorld.jar Main.java Bug.java
```

- **Exécuter** le programme (le point d'entrée est dans la classe *Main*). Les données *murs* et *biscuits* sont lues depuis le fichier *mazeData* :

```
java -cp .:BugWorld.jar Main < mazeData.txt
```

- **Modifier** le code source de la méthode **enRoute()** pour réaliser l'objectif, en n'utilisant que les méthodes de déplacement `avance()`, `droite()` et `gauche()` puis la méthode `prendBiscuit()`.
- **Compiler**, puis **exécuter** de nouveau pour vérifier la bonne réalisation.

> ***Note***
>
> Au fil des exécutions de votre programme, surveiller les messages affichés dans la console.

Second défi : reproduire un motif

Les objectifs de cet exercice sont :

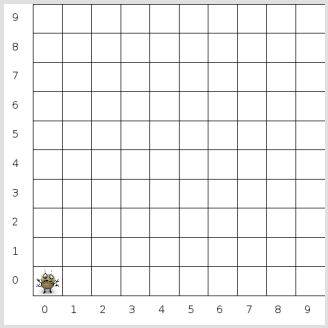
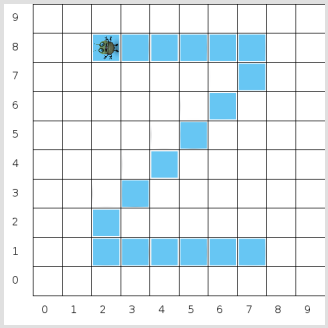
- Dessiner des motifs colorés sur la grille, en utilisant la brosse du Bug.
- **Commenter** le code source.

Pour tracer le motif, il faut alternativement lever et baisser la brosse du Bug. La couleur est au choix ; celle de l'exemple est *StdDraw.BOOK_LIGHT_BLUE*. La liste des couleurs prédéfinies est accessible dans la documentation de la bibliothèque *StdDraw*, ici.

Enfin, pour positionner directement le Bug dans la case où il doit débuter son tracé, on utilisera les méthodes *setX()* et *setY()*.

> ***Note***
>
> Comme il n'y a ici ni mur ni biscuit, plutôt que de fournir en entrée en fichier *mazeData.txt* vide, on va **ajuster** la variable `readMaze` dans la classe *Main*, comme ceci

```
boolean readMaze = false ;
```

État initial	Résultat attendu
• Un Bug dans un monde de 10x10 cases.	• Le Bug s'est déplacé en traçant un **Z**.
	

À faire

- **Compléter** le code la méthode `enRoute()` de sorte à dessiner le motif **Z** attendu. **Commenter** le code au fur et à mesure, de manière pertinente, c'est à dire ni trop ni trop peu.
- **Modifier** le code pour que le Bug revienne à sa case de départ en *effaçant* ses traces.
- **Modifier** votre code pour que le Bug Trace cette fois une lettre **A** de couleur verte, sauf sa barre horizontale qui sera jaune.

l'API du monde des Bugs

Note

- API : Application Programming Interface

Vous trouverez ci-dessous la description de tout ce peuvent faire les Bugs. Chaque action sera effectuée lorsque vous appelerez la *méthode* associée.

Un Bug peut bouger

Action	Méthode	Commentaires
Avancer	void avance()	Avance d'un pas dans la direction du Bug.
Avancer	void avance(int nbPas)	Avance de *nbPas* pas dans la direction du Bug.
Tourner à droite	void droite()	Tourne de 90° dans le sens horaire.

Tourner à gauche	void gauche()	Tourne de 90° dans le sens anti-horaire.
Se retourner	void retourne()	Tourne de 180°.
Fixer l'orientation	void setDirection()	Le paramètre est une chaine ("N", "S", "E" ou "O").
changer d'abscisse	void setX(int x)	Se place dans la colonne d'indice *x* (ligne inchangée).
Changer d'ordonnée	void setY(int y)	Se place dans la ligne d'indice *y* (colonne inchangée).
Fixer la vitesse de déplacement	void setVitesse(double v)	Fixe la vitesse à **v** mouvements par seconde (approximativement).

Un Bug peut s'informer

Action	Méthode	Commentaires
Obtenir l'abcisse	int getX()	La valeur renvoyée est l'indice de la colonne du Bug
Obtenir l'ordonnée	int getY()	La valeur renvoyée est l'indice de la ligne du Bug
Obtenir l'orientation	String getDirection()	La valeur renvoyée est une chaine ("N", "S", "E" ou "O").
Obtenir la couleur du sol	Color getCouleurSol()	La valeur renvoyée est la couleur du sol à l'endroit du Bug.
Obtenir la couleur de brosse	Color getCouleur()	La valeur renvoyée est la couleur laissée au sol par le Bug quand il se déplace avec la brosse baissée.
Obtenir l'état de la brosse	boolean isBrosseBaisse()	Renvoie *true* si la brosse est baissée.
Y a-t-il un biscuit ?	boolean isSurBiscuit()	Renvoie *true* si un biscuit est sur la même case que le Bug.
Suis-je face à un mur ?	boolean isFaceMur()	Renvoie *true* si le Bug fait face à un mur intérieur ou à un bord de grille.

Un Bug peut interagir

Action	Méthode	Commentaires
Changer la couleur de brosse	void setCouleur(Color c)	Adopte la couleur *c*.
Baisser la brosse	void baisseBrosse()	Permet de colorier les cases visitées.
Lever la brosse	void leveBrosse()	Permet de ne pas altérer la couleur des cases visitées.

Prend un biscuit	void prendBiscuit()	Ramasse un biscuit qui se trouverait sur la case du Bug. Un Bug peut porter plusieurs biscuits.
Pose un biscuit	void poseBiscuit()	Pose un biscuit sur la case du Bug.
Émettre un message	void message(String s)	Affiche la chaine *s* dans la console