Programmation 1

Christian Grévisse (christian.grevisse@uni.lu)
Gilles Neyens (gilles.neyens@uni.lu)

TP 2 – Types de données, Structures de décision & *Code Smells*

1 Types de données

Dans cette partie du TP vous n'avez pas le droit d'utiliser les structures de décision que vous avez appris (`if-then(-else)`, `switch/case` statements ou `? :`).

Exercice 1 – Dis-moi la vérité!
L'opérateur d'implication, dénoté $\rightarrow$, est défini par la table de vérité suivante:

a	b	a $\rightarrow$ b
false	false	true
false	true	true
true	false	false
true	true	true

Exprimez l'opérateur d'implication en n'utilisant que des opérateurs logiques simples, c.-à-d. *et* ($\wedge$, en Java `&&`), *ou* ($\vee$, en Java `||`) et la négation ($\neg$, en Java `!`).

Ecrivez un programme qui implante votre expression et affiche la table de vérité. Vérifiez votre résultat à l'aide de la table de vérité ci-dessus. Notez que la priorité des opérateurs logiques dans l'ordre décroissant est *négation* $>$ *et* $>$ *ou*, p.ex.

$$a \wedge b \vee \neg c \iff (a \wedge b) \vee (\neg c)$$
$$a \wedge \neg b \vee c \iff (a \wedge (\neg b)) \vee c$$

$\iff$ signifie "équivalent à".

Exercice 2 – Comparaison de deux nombres positifs
Ecrivez un programme qui lit deux nombres **positifs** de l'entrée standard (vous aurez besoin de la méthode `Math.abs` pour garantir que les nombres sont positifs). Le programme doit indiquer si le premier nombre est plus grand que le deuxième. Si c'est le cas le programme va afficher 1, et 0 sinon. Faites la différence entre les deux cas suivants:

1° Les deux nombres sont des entiers. Utilisez la division entière pour faire la comparaison.

2° Au moins un des nombres est un nombre décimal. Utilisez la division à virgule flottante pour faire la comparaison.

Les méthodes `Math.min` et `Math.floor` pourraient être utiles.

Exercice 3 – Go with the (over-)flow
En utilisant le type de donnée `byte` (plage de valeurs $[-128; 127]$), montrez qu'il y a un overflow (dépassement de capacité) dès que les limites de la plage de valeurs sont atteints. Montrez-le pour la limite inférieure et pour la limite supérieure. Qu'est-ce que vous observez et comment pouvez-vous l'expliquer?

2 Structures de décision

Exercice 4 – United States of Aggregation

Ecrivez un programme qui lit la température (en degré Celsius) de l'entrée standard et affiche si l'eau sera a l'état solide (glace), liquide ou gazeux (vapeur) à cette température.

Exercice 5 – Multiplication

Ecrivez un programme qui lit deux entiers de l'entrée standard et qui calcule le signe du produit sans calculer le produit lui-même. Le programme affichera 1 si le produit est positif et -1 si le produit est négatif. Au lieu d'utiliser la structure de décision `if-then-else`, essayez d'utiliser l'opérateur `? :`.

Opérateur conditionnel `? :`

L'opérateur conditionnel `? :` est un opérateur ternaire (trois opérandes) qui peut remplacer des instructions `if-then-else` courtes. Par exemple, le code suivant

```
1 int n;
2 if(<someCondition>) {
3   n = 42;
4 } else {
5   n = 21;
6 }
```

peut être réécrit

```
1 int n = <someCondition> ? 42 : 21;
```

ce qui va rendre votre code plus concis et lisible.

Exercice 6 – Tri

Ecrivez un programme qui lit trois nombres entiers de l'entrée standard et qui les affiche dans l'ordre croissant.

1° Trouvez d'abord une solution avec des `if` imbriqués.
2° Ensuite, essayez de trouver une solution sans imbrication.

Exercice 7 – Attendez une seconde ...

1° Ecrivez un programme qui lit l'heure (heures, minutes et secondes) de l'entrée standard. L'utilisateur pourra aussi spécifier le format de l'heure (format à 12 heures avec AM/PM ou format à 24 heures). N'oubliez pas de vérifier dans votre programme que les données entrées sont correctes (p.ex. les minutes entrées doivent être compris entre 0 et 59).
2° Affichez le temps indiqué dans le format `hh:mm:ss` et n'oubliez pas de mettre des 0 en tête si les valeurs sont plus petites que 10.
3° Avancez l'heure par une seconde. Par exemple, si l'heure entrée est `09:59:59`, la nouvelle heure sera `10:00:00`.

4° Affichez la nouvelle heure.

Exercice 8 – Rectangles

Ecrivez un programme qui détermine si un point $P(x_P; y_P)$ se trouve é l'intérieur d'un rectangle. Le programme lit les coordonnées de P de l'entrée standard. Il va aussi lire les coordonnées du rectangle de l'entrée standard. Un rectangle peut être définit par quatre coordonnées $(x_{min}, y_{min}, x_{max}, y_{max})$ comme on ne prend en compte que les rectangles dont les côtés sont parallèles aux axes.

ℹ Un test pareil peut être utilisé dans le développement de jeux vidéo pour détecter des collisions entre des sprites (Partie graphique d'un objet).

Exercice 9 – Don avec contrepartie

Sur des platformes de crowdfunding comme Kickstarter ou Indiegogo, il est courant d'offrir des récompenses aux gens qui financent des projets en faisant des dons.

Dans cet exercice, l'utilisateur doit pouvoir entrer le montant en euros qu'il veut contribuer. Les montants possibles sont 10 €, 20 €, 50 €, 100 €, 200 € et 500 €.

Les récompenses suivantes sont données aux supporteurs d'un projet:

10 €	Un High five avec les assistants
20 €	Votre nom en ASCII
50 €	Affichage public de votre don sur Moodle
100 €	– Une clé USB de 8Go – Affichage public de votre don sur Moodle
200 €	– Couvercle Uni.lu pour votre Smartphone – Une clé USB de 8Go – Affichage public de votre don sur Moodle
500 €	– Autographe de vos professeurs – Couvercle Uni.lu pour votre Smartphone – Une clé USB de 8Go – Affichage public de votre don sur Moodle

En utilisant la structure `switch`, affichez ces récompenses. Essayez de minimiser les appels à la méthode `System.out.println` pour chaque montant. Si l'utilisateur entre un montant qui n'est pas contenu dans le tableau, affichez un message d'erreur.

3 Styles de code & Code Smells

Exercice 10 – Smelly cat, smelly cat, ...
Vous souvenez-vous de Phoebe de la sitcom *Friends*? Sa chanson *Smelly Cat* fournit un parfait example pour un code smell!

Observez le code suivant:

```
1     String smellyDog = "smelly cat";
2     System.out.println(smellyDog + ", " + smellyDog + ", what are they feeding you?");
3     System.out.println(smellyDog + ", " + smellyDog + ", it's not your fault!");
```

Le code a l'air bien et affiche tout correctement. Cependant il y a un smell sémantique dans le code comme le nom de la variable `smellyDog` est incorrecte. Bien que le nom d'une variable ne va pas changer l'exécution d'un programme, il peut être la cause de confusions si le programme va être développé d'avantage dans le futur. En plus cette variable apparaît 5 fois, la renommer peut donc conduire à des erreurs si on oublie de renommer toutes les utilisations de la variable.

C'est pourquoi Eclipse a des options de refactorisation dont fait partie l'option de renommer une variable. Faites un clic droit sur le nom de la variable que vous voulez renommer et choisissez `Refactor` → `Rename` .... Maintenant toutes les occurrences de la variable apparaissent avec une bordure autour et vont être changées en même temps que vous tapez le nouveau nom. Quand vous avez fini appuyez sur Entrée. Les 5 occurrences de la variable devraient maintenant être changées en `smellyCat` au lieu de `smellyDog`.

Exercice 11 – `goto fail`
Vous avez vu dans le cours magistral que la syntaxe des structures `if` en Java vous donne la possibilité d'omettre les accolades dans le cas où il y a seulement une instructions après le if.

Qu'est-ce que vous pensez du pseudo-code suivant? La `conditionB` va-t-elle être évaluée? Pourquoi (pas)? Quels problèmes de sécurité peuvent apparaître? Quelles conséquences cela peut avoir pour le développement de nouveau langages de programmation?

```
1 if(<conditionA>)
2   stopProgram();
3   stopProgram();
4 if(<conditionB>)
5   stopProgram();
```

↯ Pour savoir plus sur ce sujet lisez un article sur le *goto fail bug*, une partie de code inaccessible dans l'implémentation SSL/TLS de Apple, découvert début 2014.

Exercice 12 – Save Actions
Lisez le document sur les *Save Actions* sur Moodle pour configurer votre IDE Eclipse.