Nom : N° Etudiant:

Programmation bas niveau - L2 informatique

Premier contrôle continu écrit

Mardi 13 Novembre 2012 (1 heure – aucun document autorisé)

Première Partie: QCM (8 points)

Cochez les bonnes réponses *(Bonne réponse: 1 point ; mauvaise réponse: -1 point ; pas de réponse: 0)*

1. Quelles sont les avantages de la machine virtuelle de type JVM ?
   - A- Portabilité
   - B- Contrôle direct du matériel
   - C- Vitesse d'exécution
   - D- Facilité des mises à jour
2. Quelle est la commande pour passer d'un **.class** à un **.j** ?
   - A- javap -c nomDeFichier
   - B- java -jar jasmin.jar nomDeFichier.j
   - C- java nomDeFichier
   - D- java -jar nomDeFichier
3. Quelle est la commande pour passer d'un **.j** à un **.class**?
   - A- javap -c nomDeFichier
   - B- java -jar jasmin.jar nomDeFichier.j
   - C- java nomDeFichier
   - D- java -jar nomDeFichier
4. A quoi sert **.limit stack 99** ?
   - A- Augmenter la grandeur de la pile à 99
   - B- Augmenter la grandeur du tableau de variables à 99
   - C- Augmenter le stockage à 99 fichiers
   - D- Augmenter la consommation mémoire de la JVM à 99
5. A quoi sert **.limit locals 99** ?
   - A- Augmenter la grandeur de la pile à 99
   - B- Augmenter la grandeur du tableau de variables à 99
   - C- Augmenter le stockage à 99 fichiers
   - D- Augmenter la consommation mémoire de la JVM à 99
6. Marquez les opérations correctes:
   - A- bipush 300
   - B- sipush 30
   - C- lcd 3.14
   - D- iadd 3
7. Marquez les opérations correctes:
   - A- dup
   - B- fmul 4.0 5.0
   - C- pop
   - D- swap
8. A quoi sert le i de istore ?
   - A- C'est une notation issue de la programmation iOS (Apple)
   - B- Pour montrer la différence entre les types de magasins
   - C- A rien car le tableau de variables peut très bien contenir des float, int, String, etc.
   - D- Pour préciser le type de pile utilisé

Deuxième (2 points) Expliquez comment fonctionne la pile de la JVM en 3-4 lignes

Nom : N° Etudiant:

Troisième partie: Analyse (5 points)

Analysez les programmes suivants. S'il y a des erreurs, expliquez et corrigez-les, sinon expliquez le déroulement des programmes (c'est à dire l'état de la pile) en vous aidant des numéros de lignes.

#	Programme 1	Programme 2
01	.class public prg1	.class public prg1
02	.super java/lang/Object	.super java/lang/Object
03		
04	.method public static main([Ljava/lang/String;)V	.method public static main([Ljava/lang/String;)V
05	.limit stack 5	.limit stack 5
06		
07	getstatic java/lang/System/out Ljava/io/PrintStream;	getstatic java/lang/System/out Ljava/io/PrintStream;
08	bipush 4	ldc 38313
09	bipush 3	dup2
10	idiv	ldc 3600
11	ldc 3.141593	idiv
12	fmul	invokevirtual java/io/PrintStream/println(I)V
13	bipush 5	
14	dup	ldc 3600
15	dup	irem
16	imul	dup2
17	imul	bipush 60
18	fmul	idiv
19	invokevirtual java/io/PrintStream/println(F)V	invokevirtual java/io/PrintStream/println(I)V
20		
21	return	bipush 60
22	.end method	irem
23		invokevirtual java/io/PrintStream/println(I)V
24		
25		return
26		.end method

Nom : N° Etudiant:

Quatrième partie: Programmation (5 points)

Ecrivez un programme (commenté) jasmin qui affiche dans l'ordre croissant tous les nombres impairs compris entre 1 et 200.

Nom : N° Etudiant:

Pour avoir le bytecode d'un fichier .class:

`javap -c nomDeFichier`

-c : pour avoir le détail des méthodes

nomDeFichier : un fichier .class (ne pas ajouter l'extension)

Pour compiler un fichier .j (jasmin) en .class:

`java -jar jasmin.jar nomDeFichier.j`

Pour exécuter ce fichier .class sur une JVM

`java nomDeFichier` (ne pas ajouter l'extension)

Squelette le plus simple d'un fichier .j

```
.class public Maclasse
.super java/lang/Object
.method public static main([Ljava/lang/String;)V
   .limit stack 99
   .limit locals 99    ; Si on utilise des variables
   ; VOTRE CODE ICI
   return
.end method
```

Exemple pour afficher un entier :

```
getstatic java/lang/System/out Ljava/io/PrintStream;
bipush 4 ; Par exemple
invokevirtual java/io/PrintStream/println(I)V
```

Exemple pour afficher la lettre A:

```
getstatic java/lang/System/out Ljava/io/PrintStream;
ldc 65 ; équivalent de «A» en unicode
invokevirtual java/io/PrintStream/println(C)V
```

Exemple pour afficher un «coucou»

```
getstatic java/lang/System/out Ljava/io/PrintStream;
ldc "coucou"
invokevirtual java/io/PrintStream/println(Ljava/lang/String;)V
```

Définition d'une méthode générique (à faire en dehors du main):

```
.method public static nomMethod(type_paramètres)type_résultat
   .limit stack 99
   .limit locals 99
    ; VOTRE CODE
   return
.end method
```

`type_résultat` et `type_paramètres` sont de type: F, V, I , FF... (si aucun paramètre d'entrée, mettre **`()`**)
Par exemple, la méthode **`rajoute(II)I`** va prendre 2 entiers en paramètres (que l'on accédera dans la méthode grâce à `iload 0` et `iload 1`) et renverra un entier (avec `ireturn`).

Appel de la méthode

Si besoin, placez les paramètres au sommet de la pile, puis utilisez

`invokestatic nomClass/nomMethod(type_paramètres) type_résultat`

Le résultat (s'il y en a un) est stocké au sommet de la pile, en remplacement des paramètres. Dans l'exemple précédent, il faudra 2 entiers sur la pile, puis on fait un **`invokestatic Maclasse/rajoute(II)I`**, ce qui supprimera ces deux entiers de la pile et mettra un autre (le retour de la méthode) sur la pile.