Correction du devoir 2

1 Connaissance de Java

Pour chaque élément cité, n'oublier pas de donner des précisions (voir cours p.ex.).

1.1 Différents types d'applications que l'on peut réaliser en Java

Ce sont :
- les applets
- des applications indépendantes (stand alone)
- des serveurs web

1.2 Force de Java

Les caractéristiques qui font la force de Java sont :
- la fiabilité
- la portabilité
- la lisibilité
- l'orientation Objet
- la rapidité de programmation
- la sécurité

Cette liste reprend les idées les plus importantes, elle n'est bien entendu pas exhaustive.

2 Format de données

2.1 Questions

2.1.1 Longueur d'un tableau

Elle est obtensible par la variable tab.length. Cette valeur est égale à la valeur de l'indice maximum du tableau + 1.

2.1.2 Affichage du contenu d'un StringBuffer à l'écran

Le code est :
System.out.println(buf.toString()) ;
ou plus simplement System.out.println(buf) ;

2.1.3 Classe abstraite

Classe qui ne peut être instanciée, et qui défini un certain nombre d'états (les attributs de classe) et de comportement (les méthodes), qui peuvent être héritées. Certaines méthodes peuvent être abstraites, dans ce cas elles doivent obligatoirement être instanciées dans les méthodes filles.

2.1.4 déclaration, l'initialisation et l'instanciation

déclaration : c'est l'annonce de l'existance d'une variable d'un nom et d'un type donné. Ex : String nom ;

instanciation : c'est la création d'un nouvel objet à partir d'une classe. Ex : new ObjetToto() ;

initialisation : c'est l'attribution d'une instance à une variable.On peut donc accéder au contenu de l'objet par cet variable. Ex : nom = "Nicolas" ;

2.1.5 Diagramme de classe UML

voir cours. Les notions à préciser au minimum sont : package, classe, attributs de classe, méthodes.

2.1.6 Diagramme de séquence UML

voir cours. Les notions à préciser au minimum sont : Objet/Instance, ligne de vie, communication par message, terminaison.

2.1.7 Parcours de tableau

Le code correspondant est :

```
public class Compte{

    public static void compteParN(int N){
if(N<=0||N>100) System.out.println("Erreur, N doit être compris
entre 0 exclus et 100 inclus");
else{
    int i=0;
    while(i*N<=100)
System.out.println("compte = "+i++*N);
}
    }

    public static void main(String[] args){

compteParN(3);
    }
}
```

3 Code sur Machine

3.1 Eleveur

Code de la classe Eleveur.java :

```
package virtuel;

public class Eleveur{
```

```
    String nom;
    String nomAnimal;

    public Eleveur(String nom, String nomAnimal){

this.nom = nom;
this.nomAnimal = nomAnimal;

    System.out.println("Bonjour, je m'appelle "+nom+" et mon
tamagotchi s'appelle "+ nomAnimal);
    }


    public static void main(String[] args){

Eleveur jacquot = new Eleveur("jacquot","toto");
Potager jardin = new Potager(50);
Tamagotchi toto = new Tamagotchi(jacquot.nomAnimal, jardin);
try{
    toto.vivre();
}
catch(FaimException fe){
    System.out.println("Pauvre Tamagotchi, il n'a plus a manger,
il va mourir. Sniiifff.");
    toto.dernierSoupir();
    toto = null;
}
jacquot.attendre();
}

    public void attendre(){
System.out.println("Je suis "+ nom+" et je laisse mon Tamagotchi
 vivre comme un grand");
    }

}
```

3.2 Potager

Code de la classe Potager.java :

```
package virtuel;

public class Potager{

    int contenu;

    public Potager(int stock){
contenu = stock;
```

```
System.out.println("Potager créé");
    }

    public void manger(int i) throws FaimException{

contenu-= i;
if(contenu<=0) throw new FaimException();
    }

    public int inventaire(){
return contenu;
    }
}
```

3.3 Tamagotchi

Code de la classe Tamagotchi.java :

```
package virtuel;

public class Tamagotchi{

    String nom;
    Potager monJardin;

    public Tamagotchi(String prenom, Potager terre){
nom = prenom;
monJardin = terre;
System.out.println("Je suis "+nom+" le Tamagotchi, et je vis dans
un jardin");
    }

    public void vivre() throws FaimException{

System.out.println(nom+" : \"Enfin libre !\"");

for(int i= 0; i < 12;i++){
    monJardin.manger(10);
    System.out.println("il reste : "+ monJardin.inventaire() +"
kilo(s) de nourriture");
}

System.out.println("il reste : "+ monJardin.inventaire() +
" kilo(s) de nourriture");

    }

    public void dernierSoupir(){
System.out.println("Je suis "+nom+" le Tamagotchi et je chante, je
chante, je chante le chant du cygne \n AhhhhhhhAhhhhhAhhhAhAha");
```

```
    }

}
```

3.4 FaimException

Code de la classe FaimException.java :

```
ackage virtuel;

public class FaimException extends Exception{
}
```