1 Serialisation Simple

...

2 Gestion des objets nuls

Soit l'interface

```
public interface Creator<T extends MySerialisable>{
  public abstract T init();
}
```

Question 1 Ajouté un champ **public static final** Creator CREATOR aux classes Test1 et Test2 qui appel le constructeur vide de la classe.

Question 2 Ajouté à SerializerBuffer deux méthodes :

```
  public void writeMySerialisable( MySerialisable );
  public <T extends MySerialisable> T readMySerialisable( Creator<T> );
```

La première écrit l'objet sur le buffer en appelant sa méthode writeToBuff(SerializerBuffer). la seconde crée un objet du type T grâce au Creator<T>, puis le remplis avec les valeurs contenues dans le buffer grâce à la méthode de T readFromBuff(SerializerBuffer)

Modifier les classes Test1 et Test2 Afin qu'elle utilise ces méthodes pour la sérialisation.

Question 3 Modifier les deux méthodes de la question précédente pour gérer le cas des objets **null**.

3 Gestion des structures récursives circulaires

Question 4 Ajouté à la classe SerializerBuffer les champs suivant :

```
  private Map<Integer,Integer> objMap = new HashMap<>();
  private ArrayList<MySerialisable> objArray = new ArrayList<>();
```

Question 5 Modifier la méthode writeMySerialisable pour qu'elle enregistre tous les objets qu'elle sérialise dans le tableau objArray. On utilisera de plus System.identityHashCode(Object) pour générer un identifiant unique de chacun des objets que l'on ajoutera à la table objMap avec la position de l'objet dans le tableau.

Question 6 Modifier les méthodes writeMySerialisable() et readMySerialisable() pour gérer les structures récursives circulaires.

4 Utilisation

Soit la classe BinaryGraph qui implémente des graphes ou chaque noeud à deux successeurs. Cette classe est munie d'une méthode sum(**int** i) qui calcule la somme des clés des chemins de taille i.

```
public class BinaryGraph {
  BinaryGraph lhs,rhs;
  int key;

  public BinaryGraph(BinaryGraph lhs, BinaryGraph rhs, int key) {
    this.lhs=lhs;
    this.rhs=rhs;
    this.key=key;
  }

  public int sum(int k) {
    if(k<0)return 0;
    if(k==0)return key;
    int s = key;
    if(lhs != null)s+= lhs.sum(k-1);
    if(rhs != null)s+= rhs.sum(k-1);
    return s;
  }
}
```

Question 7 Modifier cette classe pour qu'elle implémente l'interface MySerialisable.

Question 8 Écrire un serveur qui attend des connexions sur un port. Lorsqu'un client de connecte, le serveur tente de lire un graphe binaire suivit d'un entier i sur le socket. En cas de succès, il appelle la méthode sum(i) sur ce graphe, renvoie le résultat au client puis ferme la connexion. Enfin, il se met à attendre une nouvelle connexion.