Al-JazeeraUniversity
Faculty of Engineering
Department of Informatics

جامعة الجزيرة الخاصة
كلية الهندسة
قسم الهندسة المعلوماتية

الاختبار الثاني – مقرر : "برمجة غرضية التوجه"

السؤال الأول (15 علامة) :

يسمح الصف moneyManager بمساعدة آلات البيع الأتوماتيكية بإعادة النقود للزبائن. يدير هذا الصف مخزناً من القطع النقدية. و يعيد هذا الصف المبلغ المطلوب بأقل عدد ممكن من القطع النقدية (بمعنى آخر يعيد القطع النقدية ذات القيم الأكبر قدر المستطاع).

مخزن النقود (moneyStore) هو عبارة عن (attribute) صفة من النوع TreeMap في الصف moneyManager.

يسمح الصف Money بنمذجة أنواع القطع النقدية في بلد ما. إن أغراض هذا الصف هي مفاتيح (keys) لمخزن النقود (moneyStore). بينما القيمة المرتبطة بكل مفتاح تمثل عدد القطع النقدية المتاحة في آلة البيع من قيمة هذا الغرض.

1 - لماذا تم اختيار الصفة moneyStore من النوع Map؟ **(1 علامة)**

2 - لماذا تم اختيار الصفة moneyStore من النوع TreeMap و ليس HashMap؟ **(1 علامة)**

3 - ماذا يجب أن تنفذ الأغراض التي تمثل مفاتيح الـTreeMap؟ **(1 علامة)**

4 - اكتب الصف Money و الذي يملك منهجاً ستاتيكياً لإنشاء القطع النقدية createMoney(int value). يتم استخدامه في حالة بلد كالجمهورية العربية السورية لإنشاء القطع النقدية من فئة 1، 2، 5، 10، 25. اكتب هذا الصف و المنهج createMoney بحيث يتم تحديد الترتيب بين القطع النقدية المختلفة التي يقوم بإنشائها؟ **(3 علامات)**

5 - اكتب الصف MoneyManager مع منهجين :

a. addMoney(Money m, int nbMoney) و الذي يسمح بتعديل قيمة عدد القطع النقدية المتاحة في الآلة من فئة نقدية معينة. **(4 علامات)**

b. returnMoney(int amount) و الذي يعيد المبلغ amount بأقل عدد من القطع النقدية المتاحة في MoneyStore. **(5 علامات)**

انتهت الأسئلة

بالتوفيق والنجاح د. باسم قصيبة

Appendix of TreeMap Class

public boolean **containsKey**(Object key)

public Collection **values**()

public Object **get**(Object key)

public Object **put**(Object key, Object value)

public void **putAll**(Map t) copies all of the mappings from the specified map to this map (optional operation).

public Set **keySet**() returns a Set view of the keys contained in this map. The set's iterator will return the keys in ascending order (according to compareTo).

أحد الحلول المقبولة للمسألة.

```
class Money implements Comparable {
	public final int value;
	private Money(int value){
		this.value = value;
	}

	public static Money createMoney(int value){
		return new Money(value);
	}

	public boolean equals(Object o){
		return value == ((Money) o).value;
	}
	public int hashCode(){
		return new Integer(value).hashCode();
	}

	//inverse ordering
	public int compareTo(Object o){
		Money another = (Money) o;
		return new Integer(another.value).compareTo(new Integer(this.value));
	}

}
```

```
public class MoneyManager{
	private TreeMap moneyStore;
	public MoneyManager() { moneyStore= new TreeMap();}

	public void addMoney(Money m, int nbMoney){
		if (moneyStore.containsKey(m){
			moneyStore.put(m, new Integer(((Integer) moneyStore.get(m)).intValue() + nbMoney));
		}
		else {
			moneyStore.put(m, new Integer(nbMoney));
		}

	}

	public List returnMoney(int amount) {
		List result = new ArrayList();
		while(amount >0){
			if(amount >= 25 and ((Integer) moneyStore.get(newMoney(25))intValue() >1){
				Integer x  = (Integer) moneyStore.get(newMoney(25));
				Int i = x.intValue();
				i  = i-1;
				Integer x2 = new Integer(i);
				moneyStore.put(new Money(25), x2);
				amount = amount -25;
			}
			else if(amount >= 10 and ((Integer) moneyStore.get(newMoney(10))intValue() >1){
				….
			}
			elseif(amount >= 5 and ((Integer) moneyStore.get(newMoney(5))intValue() >1){
				….
			}
			elseif(amount >= 2 and ((Integer) moneyStore.get(newMoney(2))intValue() >1){
				….
			}
			elseif(amount >= 1 and ((Integer) moneyStore.get(newMoney(1))intValue() >1){
				….
			}

		}
	}

}
```