Institiúid Teicneolaíochta, Trá Lí
INSTITUTE OF TECHNOLOGY TRALEE

AUTUMN EXAMINATION AY 2013/2014

Object Oriented Programming 3

CRN: 43853
Module Code : PROG 61005

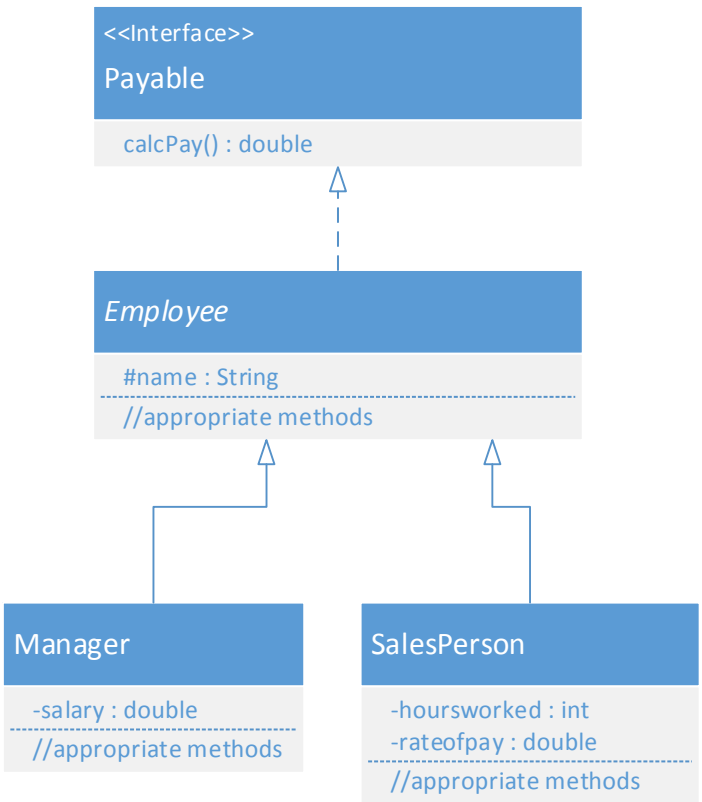
Internal Examiner: Mr. J. Walsh
External Examiner: Mr. Michael Godley

Duration of Exam: **2 Hours**

Instructions to Candidates: **Attempt two questions**

Question 1

In the VOPC diagram below, Payable is an interface, Employee is an abstract class, Manager and Salesperson are concrete classes.



a) Write the full java code for the payable interface. (4 marks)

b) Write the code, including header, attributes, abstract and other appropriate methods for the abstract *Employee* class. (10 marks)

c) Write the code for a driver class that creates a collection of Employees. No duplicates are allowed in the collection. Any inputs are taken into the application via input dialog boxes. Use an appropriate structure to allow for multiple Employees (Managers or SalesPersons) to be input by the user. (20 marks)

d) Write the code to create a filter that displays all employees whose calculated pay exceeds 500. Use an Iterator object to scan through the employee collection. (16 makes)

Question 2

a) Write a brief note on the java collection framework explaining the role of interfaces, implementations and algorithms within it. Include a diagram indicating the relationship between the various interfaces and implementations. Include the Map interface in your answer. (10 marks)

b) Compare and contrast a TreeSet with a HashSet in terms of duplicates, adding/ removing elements, speed of access and how elements are grouped together. (8 marks)

c) Write the code to populate a HashMap of Managers based on the Manager class given in the diagram above. You can assume a 2-argument constructor exists for the Manager class that takes a name and salary as arguments. The Map entry has a key corresponding to a Manager ID code (String) and a value corresponding to a Manager object. (15 marks)

d) Add additional code to part c) that will output all the Manager ID codes in the map, ask the user to enter a Manager ID code, scan the Map and display the details of the corresponding Manager. (17 marks)

Question 3

a) Write a brief note explaining polymorphism as used in Java. Include, using an example, an explanation of dynamic method binding. (10marks)

b) Explain, giving two examples of each, the difference between checked and unchecked Exceptions in Java. (4 marks)

c) Explain briefly how Exceptions are generated in Java. (4 marks)

d) Explain briefly how exceptions are handled in Java. Differentiate between handling checked and unchecked Exceptions in your explanation. (12 marks)

e) Write the code for a method called setSalary that requests a salary value of type double from the user using an input dialog box. The method should throw and handle locally, an IllegalArgument exception if the user enters a salary value less than 2000.The method should also handle a NumberFormatException if the user enters text instead of a number. (15 marks)

f) What, if any, changes would you make to the code written for part e) to further improve its robustness. (5 marks)

Appendix : Summary of Collection Interface methods

Method Summary	
boolean	**add**(E e) Ensures that this collection contains the specified element (optional operation).
boolean	**addAll**(Collection<? extends E> c) Adds all of the elements in the specified collection to this collection (optional operation).
void	**clear**() Removes all of the elements from this collection (optional operation).
boolean	**contains**(Object o) Returns `true` if this collection contains the specified element.
boolean	**containsAll**(Collection<?> c) Returns `true` if this collection contains all of the elements in the specified collection.
boolean	**equals**(Object o) Compares the specified object with this collection for equality.
int	**hashCode**() Returns the hash code value for this collection.
boolean	**isEmpty**() Returns `true` if this collection contains no elements.
Iterator<E>	**iterator**() Returns an iterator over the elements in this collection.
boolean	**remove**(Object o) Removes a single instance of the specified element from this collection, if it is present (optional operation).
boolean	**removeAll**(Collection<?> c) Removes all of this collection's elements that are also contained in the specified collection (optional operation).
boolean	**retainAll**(Collection<?> c) Retains only the elements in this collection that are contained in the specified collection (optional operation).
int	**size**() Returns the number of elements in this collection.
Object[]	**toArray**() Returns an array containing all of the elements in this collection.
<T> T[]	**toArray**(T[] a) Returns an array containing all of the elements in this collection; the runtime type of the returned array is that of the specified array.

Summary of java.util Interface Map<K,V> methods

Method Summary	
void	**clear**() Removes all of the mappings from this map (optional operation).
boolean	**containsKey**(Object key) Returns true if this map contains a mapping for the specified key.
boolean	**containsValue**(Object value) Returns true if this map maps one or more keys to the specified value.
Set<Map.Entry<K,V>>	**entrySet**() Returns a Set view of the mappings contained in this map.
boolean	**equals**(Object o) Compares the specified object with this map for equality.
V	**get**(Object key) Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.
int	**hashCode**() Returns the hash code value for this map.
boolean	**isEmpty**() Returns true if this map contains no key-value mappings.
Set<K>	**keySet**() Returns a Set view of the keys contained in this map.
V	**put**(K key, V value) Associates the specified value with the specified key in this map (optional operation).
void	**putAll**(Map<? extends K,? extends V> m) Copies all of the mappings from the specified map to this map (optional operation).
V	**remove**(Object key) Removes the mapping for a key from this map if it is present (optional operation).
int	**size**() Returns the number of key-value mappings in this map.
Collection<V>	**values**() Returns a Collection view of the values contained in this map.

java.util

Interface Map.Entry<K,V>

All Known Implementing Classes:

AbstractMap.SimpleEntry, AbstractMap.SimpleImmutableEntry

Enclosing interface:

Map<K,V>

```
public static interface Map.Entry<K,V>
```

A map entry (key-value pair). The `Map.entrySet` method returns a collection-view of the map, whose elements are of this class. The *only* way to obtain a reference to a map entry is from the iterator of this collection-view. These `Map.Entry` objects are valid *only* for the duration of the iteration; more formally, the behavior of a map entry is undefined if the backing map has been modified after the entry was returned by the iterator, except through the `setValue` operation on the map entry.

Since:

1.2

See Also:

`Map.entrySet()`

Method Summary

`boolean`	**`equals`**`(Object o)` Compares the specified object with this entry for equality.
`K`	**`getKey`**`()` Returns the key corresponding to this entry.
`V`	**`getValue`**`()` Returns the value corresponding to this entry.
`int`	**`hashCode`**`()` Returns the hash code value for this map entry.
`V`	**`setValue`**`(V value)` Replaces the value corresponding to this entry with the specified value (optional operation).