Institiúid Teicneolaíochta, Trá Lí
INSTITUTE OF TECHNOLOGY TRALEE

AUTUMN EXAMINATIONS AY 2014 - 2015

Object Oriented Programming 3

Module Code: PROG61005
CRN 43853

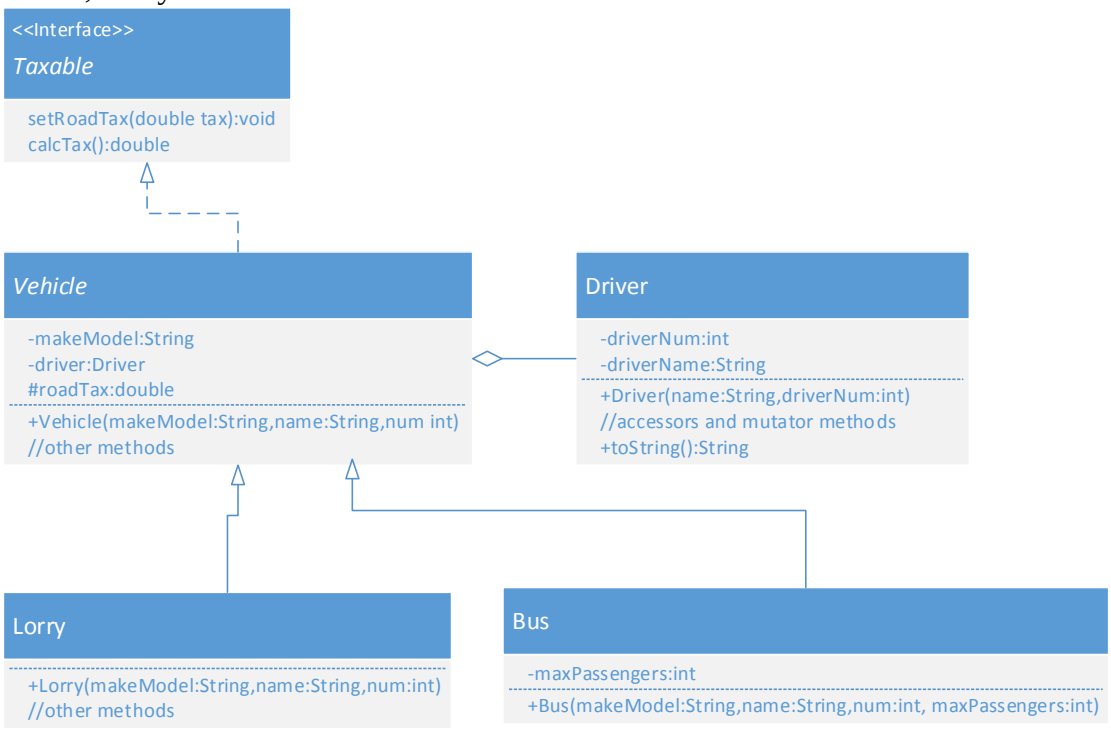
Internal Examiner: Mr. John Walsh
External Examiner: Mr Michael Godley

Duration of Exam: 2 Hours

Instructions to Candidates: **Answer Question 1 and one other question. Each question is 50 marks**

Question 1

In the UML diagram below, Taxable is an Interface, Vehicle is an Abstract Class, Driver, Lorry and Bus are concrete instantiable classes.



a) Compare and contrast an Interface, an abstract class and a concrete class. (6 marks)

b) Write the full code for the *Taxable* interface. (5 marks)

c) Write the full code, including the class header, attributes and all appropriate methods for the Vehicle class. The code should include accessor and mutator methods as well as two appropriate abstract methods and a toString() method. (17 marks)

d) Write the full code, including the class header, attributes and all appropriate methods for the Bus class. (13 marks)

e) Write the code for an application that creates a collection of Bus objects. The application should terminate when the user clicks "No" on a confirm dialog box when prompted if they would like to enter more buses. (9 marks)

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 and how elements are ordered. (6 marks)

c) Write the code to populate a HashMap of Buses based on the Bus class given in the Question 1. You can assume a makeBus() method exists that prompts the user for the bus details, creates and returns a bus object. The Map entry has a key corresponding to a busID code (String) and a value corresponding to a Bus object. (14 marks)

d) Add additional code to part c) that will output all the busID codes in the map, ask the user to enter a busID, scan the Map and display the details of the corresponding Bus. This will require you to extract a Set(s) representation of the Map. (20 marks)

Question 3

a) Consider the simple mathematical expression (3+4)*5. You can eliminate the parentheses if you write the operators after the numbers like this: 34+5*. This is known as Reverse Polish Notation and allows computation of the expression by reading the expression from left to right. An algorithm for implementing this approach using a stack is as follows:

While there is an input,
If you read an operator,
Pop two values off the stack,
Combine the two values off the stack,
Push the result back onto the stack,
Else if there is no more input,
Pop and display the result.
Else If you read a number,
Push it on to the stack,

Write the full java code to implement this algorithm. Use the Scanner Class to take in the expression one token at a time terminated with "Q". (24 marks)

b) Explain briefly how exceptions are handled in Java. (6 marks)

c) Write the code for a method called setMaxPassengers that requests a value of type int from the user using an input dialog box. The method should handle a NumberFormatException if the user enters text instead of a number. The method should also throw and handle locally, an IllegalArgument exception if the user enters a value less than 2 and greater than 200. (15 marks)

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

Appendix :

Class JOptionPane

Field Summary

Fields

Modifier and Type	Field and Description
`static int`	`CANCEL_OPTION` Return value from class method if CANCEL is chosen.
`static int`	`YES_NO_OPTION` Type used for `showConfirmDialog`.
`static int`	`YES_OPTION` Return value from class method if YES is chosen.

Method Summary

Methods

Modifier and Type	Method and Description
`static int`	`showConfirmDialog(Component parentComponent, Object message)` Brings up a dialog with the options *Yes*, *No* and *Cancel*; with the title, **Select an Option**.

Class Scanner

Constructor Summary

Constructors

Constructor and Description
`Scanner(File source)` Constructs a new `Scanner` that produces values scanned from the specified file.
`Scanner(File source, String charsetName)` Constructs a new `Scanner` that produces values scanned from the specified file.
`Scanner(InputStream source)` Constructs a new `Scanner` that produces values scanned from the specified input stream.

Method Summary

Methods

Modifier and Type	Method and Description
`void`	`close()` Closes this scanner.

boolean	**hasNext**() Returns true if this scanner has another token in its input.
boolean	**hasNextLine**() Returns true if there is another line in the input of this scanner.
String	**next**() Finds and returns the next complete token from this scanner.
String	**nextLine**() Advances this scanner past the current line and returns the input that was skipped.
long	**nextLong**() Scans the next token of the input as a long.

Interface Collection<E>

Method Summary

Methods

Modifier and Type	Method and Description
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.

Class Stack<E>

Constructor Summary

Constructors

Constructor and Description
`Stack()` Creates an empty Stack.

Method Summary

Methods

Modifier and Type	Method and Description
`boolean`	`empty()` Tests if this stack is empty.
`E`	`peek()` Looks at the object at the top of this stack without removing it from the stack.
`E`	`pop()` Removes the object at the top of this stack and returns that object as the value of this function.
`E`	`push(E item)` Pushes an item onto the top of this stack.

Class HashMap<K,V>

Constructor Summary

Constructors

Constructor and Description
`HashMap()` Constructs an empty `HashMap` with the default initial capacity (16) and the default load factor (0.75).
`HashMap(int initialCapacity)` Constructs an empty `HashMap` with the specified initial capacity and the default load factor (0.75).
`HashMap(int initialCapacity, float loadFactor)` Constructs an empty `HashMap` with the specified initial capacity and load factor.
`HashMap(Map<? extends K,? extends V> m)` Constructs a new `HashMap` with the same mappings as the specified `Map`.

Method Summary

Methods

Modifier and Type	Method and Description

Set<Map.Entry<K,V>>	entrySet() Returns a Set view of the mappings contained in this map.
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.
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.
void	putAll(Map<? extends K,? extends V> m) Copies all of the mappings from the specified map to this map.

Interface Map.Entry<K,V>

Method Summary

Methods

Modifier and Type	Method and Description
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.

Interface Iterator<E>

Method Summary

Methods

Modifier and Type	Method and Description
boolean	hasNext() Returns true if the iteration has more elements.
E	next() Returns the next element in the iteration.
void	remove() Removes from the underlying collection the last element returned by this iterator (optional operation).