M255/Specimen

Mathematics, Computing and Technology: Level 2
M255 Object-oriented programming with Java

Time allowed: 3 hours

This examination is in **TWO** parts and you should attempt **BOTH** parts.

Part 1 carries 40% of the total exam score and contains 20 multiple-choice questions each worth 2 marks. You should answer the questions by ticking the specified number of tick boxes per question in this examination booklet. You will be penalised if you tick more boxes than specified. The left-hand pages of this part of the booklet have been left blank so that you can use them for rough working. You are advised to spend about one hour on this Part.

Part 2 carries 60% of the total exam score.
This part contains four questions each covering a block of the course. You should attempt **ALL** questions.

Write your answers to Part 2 in the answer book(s) provided, starting each question on a new page. Indicate which questions you have attempted in the spaces provided on the front cover. You are advised not to cross through any work until you have replaced it with another solution to the same question. You are advised to spend about two hours on this Part.

At the end of the examination

Check that you have completed the grid below, and written your personal identifier and examination number on *each* answer book used. **Failure to do so will mean that your work cannot be identified**.

Examination Number								
Personal Identifier								

Attach this examination paper to the front of the answer books you have used for Part 2, put your signed desk record on top and fix them all together with the fastener provided.

This question paper must NOT be removed from the examination room; you must attach it to your answer book(s) at the end of the examination.

The University reserves the right not to mark your script if you fail to follow these instructions.

This page is for rough working only.

Part 1

This part carries 40% of the total exam score. Answer all 20 questions in Part 1, each of which carries 2 marks. You are advised to spend approximately 1 hour on this part.

Question 1

Consider the following shape:

If this is a visual representation of a software object, which **two** of the following are likely attributes of such an object? (*tick two boxes*)

A. ☐ length

B. ☐ 4

C. ☐ 2

D. ☐ width

E. ☐ cm

Question 2

Which **two** of the following Java statements, when executed in the OUWorkspace, will always result in `frog1` having the same colour as `frog2`, where `frog1` and `frog2` reference instances of the `Frog` class? (*tick two boxes*)

A. ☐ `frog1.colour() = frog2.colour();`

B. ☐ `frog1.setColour = frog2.getColour;`

C. ☐ `frog1.setColour(frog2.getColour());`

D. ☐ `frog1.sameColourAs(frog2);`

E. ☐ `frog1.colour = frog2.setColour();`

This page is for rough working only.

Question 3

What do the variables `num1` and `num2` hold after execution of the following code?

```
double num1 = 9;
int num2 = 2;
num1 = num1 / 2;
num2 = (int) num1 / num2;
```

(*tick one box*)

A. ☐ `num1` holds `4` `num 2` holds `2`

B. ☐ `num1` holds `4.0` `num2` holds `2`

C. ☐ `num1` holds `4` `num2` holds `2.0`

D. ☐ `num1` holds `4.5` `num2` holds `2.25`

E. ☐ `num1` holds `4.5` `num2` holds `2`

F. ☐ An error is generated.

Question 4

A Java while loop has the following structure:

```
while (condition)
{
   statement block
}
```

Which **two** of the following statements are **true**? (*tick two boxes*)

A. ☐ `condition` is first evaluated after the statement block has been executed for the first time.

B. ☐ `condition` must evaluate to a `boolean` value.

C. ☐ `condition` is only ever evaluated once.

D. ☐ If `condition` doesn't ever evaluate to `false` an exception is thrown.

E. ☐ The `statement block` might never be executed.

F. ☐ The braces around the `statement block` must be included even if it consists of a single statement.

This page is for rough working only.

Question 5

Which **two** Java `boolean` expressions are equivalent to this expression?

```
!((count > 2) || (count <= 1))
```

(tick two boxes)

A. ☐ `(count <= 2) || (count > 1)`

B. ☐ `(count < 2) || (count >= 1)`

C. ☐ `(count >= 1) && (count < 2)`

D. ☐ `(count > 1) && (count <= 2)`

E. ☐ `!(count <= 1) && !(count > 2)`

F. ☐ `!(count < 1) && !(count >= 2)`

Question 6

Which **two** of the following statements about constructors in Java are **true**?
(*tick two boxes*)

A. ☐ The programmer can name a constructor with any valid identifier.

B. ☐ If the programmer fails to write a constructor for a class the Java Virtual Machine will throw an exception.

C. ☐ A constructor will only call the zero-argument constructor of the superclass if the code `super()` is included in the body of the constructor.

D. ☐ A class may have several constructors.

E. ☐ Instances of strings and arrays can be created without using the `new` operator in conjunction with a constructor.

F. ☐ Constructors must always have the return type of the class in which they are defined.

This page is for rough working only.

Question 7

Which **two** of the following statements are **FALSE** in Java? (*tick two boxes*)

A. ☐ An object may only be assigned to a variable whose type is exactly that of the object's type.

B. ☐ An object can be assigned to a variable whose type is a superclass of the object's type.

C. ☐ An object can be assigned to a variable whose type is a subclass of the object's type.

D. ☐ Any object can be assigned to a variable that has been declared as being of type `Object`.

E. ☐ An object can be used as an actual argument to a method where the formal argument's type is an interface type which that object's class implements.

F. ☐ Attempting to assign a value of type `double` to a variable that has been declared as type `int` results in a semantic error.

Question 8

Consider this Java declaration from the `MyClass` class definition:

```
private int myInt;
```

No getter method has been provided for `myInt`.

Which **two** of these statements about `myInt` are **true**? (*tick two boxes*)

A. ☐ `myInt` is a class variable.

B. ☐ `myInt` is accessible from all instance methods of `MyClass`.

C. ☐ `myInt` is accessible from all class methods of `MyClass`.

D. ☐ `myInt` is accessible from all instance methods of subclasses of `MyClass`.

E. ☐ In order to access `myInt` from a subclass of `MyClass` it must be prefixed by the class name, like this: `MyClass.myInt`.

F. ☐ `myInt` is accessible from any constructor of `MyClass`.

This page is for rough working only.

Question 9

Consider the following code which a programmer intends to be a statement to assign a `String` object referenced by a string literal to a variable:

```
int x = "hello world
```

Which **two** of the following statements are **true**? (*tick two boxes*)

A. ☐ There are three errors in this statement.

B. ☐ Omitting a string delimiter is a semantic error.

C. ☐ Assigning a value of type String to a variable of type int is a syntax error.

D. ☐ The statement will not compile.

E. ☐ The statement will compile but will not run.

F. ☐ There are two errors in this statement.

Question 10

Which **two** of the following statements about exceptions in Java are **true**?
(*tick two boxes*)

A. ☐ `NullPointerException` is an example of a Java built-in exception class.

B. ☐ Code will not compile if a programmer does not attempt to handle an unchecked exception.

C. ☐ The `catch` block in a `try-catch` statement is always executed.

D. ☐ When a method throws an exception it is first caught by the compiler.

E. ☐ The `try-catch` statement is used to catch dynamic semantic errors.

F. ☐ Unchecked exceptions are subclasses of the class `Exception`.

This page is for rough working only.

Question 11

The following are consecutive statements of a method that compiles successfully:

```
Map<Character, Integer> charMap = new HashMap<Character, Integer>();
Map<String, Integer> stringMap = new TreeMap<String, Integer>();
charMap.put('a', 0);
charMap.put('a', 1);
stringMap.put("num", 2);
stringMap.put("num", 3);
int sum = charMap.get('a') + stringMap.get("num");
 // other statements ...
```

Which **two** of the following statements would be **true** immediately after a value was assigned to `sum`? (*tick two boxes*)*)*

A. ☐ the value of `sum` will be `2`.

B. ☐ the value of `sum` will be `3`.

C. ☐ the value of `sum` will be `4`.

D. ☐ the value of `sum` will be `5`.

E. ☐ an exception will be caused.

F. ☐ both maps will contain a single entry.

Question 12

Which **two** of the following statements are **true** about Java 1.5? (*tick two boxes*)

A. ☐ Primitive types are valid element types for instances of `ArrayList`.

B. ☐ In some implementations of the `Set` interface, the elements of the collection are sorted.

C. ☐ `Collection` is an interface, whereas `List` is a concrete class.

D. ☐ When an element is removed from the middle of an `ArrayList` object, this will leave a gap.

E. ☐ When an attempt is made to add an `int` value to an instance of `ArrayList` that holds `Integer` objects, the `int` value will be autoboxed before it is added.

This page is for rough working only.

Question 13

Which **two** of the following statements are **true** about Java 1.5? (*tick two boxes*)

A. ☐ Any code using a `for` or a `while` loop can always be replaced with code using a `foreach` loop.

B. ☐ If an array contains objects, a `foreach` loop is suitable for changing the state of all the objects in the array.

C. ☐ Using a `foreach` loop as opposed to a `for` loop ensures that no element in a collection is accidentally missed.

D. ☐ When you want iteration through an array to stop once a desired element is found, a `foreach` loop should generally be used.

E. ☐ It is not possible to iterate over some elements of an array without accessing every element in that array.

Question 14

Which **two** of the following statements are **true** about Java 1.5? (*tick two boxes*)

A. ☐ A map can contain duplicate keys.

B. ☐ With sorted maps based on the natural ordering of the keys, the decision about whether two potential keys `a` and `b` are duplicates depends on the value of `a.compareTo(b)`.

C. ☐ If `a == b` is `true` and `equals()` has not been overridden by the class of `a` or one of its superclasses, then `a.equals(b)` will automatically return `true`.

D. ☐ If `a.equals(b)` is `true` then `a.hashcode()` will automatically return the same value as `b.hashcode()`.

E. ☐ Whenever `a.equals(b)` returns `0`, then `a.compareTo(b)` will automatically return `true`.

This page is for rough working only.

Question 15

Which **two** of the following statements are **true** about Java 1.5? (*tick two boxes*)

A. ☐ If an array referenced by the variable `test` has exactly 20 elements then code using `test[20]` will cause an `ArrayIndexOutOfBounds` exception.

B. ☐ If when appending a string to an instance of `StringBuilder`, the length of the instance exceeds its declared buffer size there will be an overrun error.

C. ☐ Arrays that hold primitives are immutable.

D. ☐ Declaring an array variable as below:

```
int[] intArray;
```

does not result in memory being allocated for any `int` elements.

E. ☐ The number of arguments that a `main()` method has may vary depending on the class in which it occurs.

Question 16

Which **two** of the following statements are **true** about maps in Java 1.5?
(*tick two boxes*)

A. ☐ The values in an instance of `HashMap` can never be of type `int`.

B. ☐ The keys of a map may be of a primitive type.

C. ☐ When the message `values()` is sent to a map, the collection returned consists of copies of the values in the map.

D. ☐ It is generally bad practice to use mutable objects as keys in a map.

E. ☐ It is generally bad practice to use mutable objects as elements in arrays.

This page is for rough working only.

Question 17

Which **two** of the following statements are **true** about Java 1.5? (*tick two boxes*)

A. ☐ For any collection that has 4 elements or more, and whose class implements the `List` interface, messages of the form `set(3, someElement)` and `add(3, someElement)` will have the same effect.

B. ☐ For any collection with a size of 4 and whose class implements the `List` interface, a message of the form `add(4, someElement)` will result in an exception which is an instance of `IndexOutOfBoundsException`.

C. ☐ For any collection whose class implements the `List` interface, messages of the form `add(0, element)` can be used to insert an element at the start of the collection.

D. ☐ `Collections.sort()` is non-destructive.

E. ☐ By convention, all classes in the Java Collections Framework have at least two constructors, one with no argument, one taking a collection as an argument.

Question 18

Which **ONE** of the following statements best describes integration testing? (*tick one box*)

A. ☐ Integration testing, in object-oriented programming terms, is defined as either a test of a single method, or a test of a single class.

B. ☐ Integration testing is testing in which software components (groups of classes), hardware components, or both (which have previously been tested in isolation) are combined and tested together to evaluate the interaction between them.

C. ☐ Integration testing checks that modifications or additions to one part of the code have not made any other part stop working properly.

D. ☐ Integration testing checks that the overall system functions correctly.

E. ☐ Integration testing checks that the system meets the needs of a customer who has commissioned the software and provided a specification of their requirements.

This page is for rough working only.

Question 19

Which **two** of the following statements are **true** about a class diagram?
(*tick two boxes*)

A. ☐ It shows the overall structure of the proposed software, illustrating the classes that will be needed.

B. ☐ It shows the state of part of the software under development at an imagined particular point in time.

C. ☐ It models the proposed software in action (i.e. at run time), showing the message-sends involved in a specific collaboration.

D. ☐ It suggests ideas for how classes *might* be related through inheritance.

E. ☐ It shows messages passing between classes.

F. ☐ It illustrates events occurring in the software over time, so can be described as a dynamic model.

Question 20

Which **two** of the following statements are **true** about a waterfall method of software development? (*tick two boxes*)

A. ☐ Extreme Programming (XP) is an example of a waterfall method.

B. ☐ Restricting the initial development to only a small subset of the requirements is a common practice.

C. ☐ Each phase is generally visited only once, and each phase is completed before the next begins.

D. ☐ It is an example of a predictive method.

E. ☐ Prototypes are often used to test and confirm ideas about what the software is required to do.

Part 2

Part 2 carries 60% of the total exam score. Answer all four questions in Part 2. You are advised to spend approximately 1¾ hours on this part which should leave you about 15 mins checking time for the whole paper.

Question 21 [10 marks]

(i) Consider this code:

```
HoverFrog hoverFrog1 = new HoverFrog();
HoverFrog hoverFrog2 = new HoverFrog();
HoverFrog hoverFrog3 = hoverFrog1;
```

How many objects exist after this code has been executed? Explain your answer. [2]

(ii) Write a public method to be added to the `Frog` class that causes the receiver `Frog` object to be given the same position as the argument which is also a `Frog` object. The method signature is `samePositionAs(Frog)`, and it does not return an answer. Your method should include a comment and the method heading. [4]

(iii) Assuming that `hoverFrog1` and `frog1` reference objects of type `HoverFrog` and `Frog` respectively, explain what happens from the point at which this message-send is executed through to the result of the message-send:

```
hoverFrog1.samePositionAs(frog1);
```

[4]

Question 22 [15 marks]

(i) `Danceable` is an interface which specifies two methods with the signatures `pirouette(int)` and `prepareToDance()`. Neither method returns an answer. Write the `Danceable` interface. [3]

(ii) `DanceableHoverFrog` is a subclass of `HoverFrog` that implements the `Danceable` interface. It has no additional instance variables or methods apart from those specified by the interface.

When sent a message of the form `pirouette(3)`, an instance of `DanceableHoverFrog` executes the number of pirouettes indicated by the message's argument. Each pirouette consists of hovering up by 1, moving to the left, hovering down by 1, and moving to the right.

When sent a `prepareToDance()` message an instance of `DanceableHoverFrog` goes directly to height 3 and moves one stone towards stone 5 before returning directly to ground level again. This is repeated until it is at stone 5.

Write the class `DanceableHoverFrog`, including a default constructor that initialises the inherited instance variables as for the superclass.

You do not have to write any comments. [10]

(iii) Interfaces and superclasses are both mechanisms for specifying common behaviour. In two or three sentences explain the differences between these two approaches. [2]

Question 23 [15 marks]

(i) Write a single method that could be added to the `HoverFrog` class to give instances of that class a natural ordering. The ordering should be determined by the natural ordering of each `HoverFrog` object's `height` instance variable, in *descending order* of height. [3]

The rest of the question assumes that such an instance method has been added to the `HoverFrog` class to give instances of that class a natural ordering.

(ii) Instances of a class called `HoverFrogPond` will be used to store and manage a collection of `HoverFrog` objects that will be initially sorted in descending order based on `height`. Write the declarations for the following private instance variables for the `HoverFrogPond` class:

- `rankNames` – declared of a type capable of referencing an array of strings.
- `rankableHoverFrogs` – declared of a type capable of referencing a list of `HoverFrog` objects.
- `finalRankings` – declared of a type capable of referencing a map, using elements of the array `rankNames` as keys, and `HoverFrog` objects as values. [3]

(iii) Write a constructor for `HoverFrogPond`. It should have two formal arguments: `externalHoverFrogs` to reference a list of `HoverFrog` objects, and `externalRankNames` to reference an array of strings.

You may assume that when the constructor is invoked all the hoverfrogs in the list given as the first actual argument of the constructor have different heights. You may also assume that the array of strings given as the second actual argument will contain elements as follows (though the exact size of the array is not known).

0	`"Highest"`
1	`"2nd highest"`
2	`"3rd highest"`
3	`"4th highest"`
	`...`

In the body of the constructor, `externalHoverFrogs` should first be sorted by natural order (using a method from an appropriate utility class) and then assigned to `rankableHoverFrogs`, while `externalRankNames` should be assigned to `rankNames`. Finally an instance of a class that implements the `Map` interface, but which has no facilities for ordering, should be created and assigned to `finalRankings`. [3]

(iv) Write a method `populateFinalRankings()` which takes no arguments and returns no value. The map referenced by `finalRankings` should be populated by taking each of the elements, in order, from both `rankNames` and `rankableHoverFrogs`, in parallel, such that:

- the keys will be the elements from `rankNames`,
- the values will be the elements from `rankableHoverFrogs`.

The number of strings in the array `rankNames` and the number of hoverfrogs in `rankableHoverFrogs` may not match – although the array referenced by `rankNames` can be assumed to have no gaps. Entries should be made into the map until the last element of either `rankNames` or `rankableHoverFrogs` is processed.

This result will be that in the map `finalRankings`, the value with the corresponding key "`Highest`" will be the `HoverFrog` object that has the greatest height, the value with the corresponding key "`2nd highest`" will be the `HoverFrog` object that has the next greatest height etc. [6]

Question 24 [20 marks]

(i) Explain in two or three sentences the advantages and disadvantages of making an object persistent by saving it to file in a serialised form (as bytes) as opposed to saving the string representation of an object's instance variables to a text file. [3]

(ii) Consider the following method comment and header defined for a hypothetical class called `ObjectIO`.

```
/**
 * Prompts the user for a pathname and then attempts to open a stream
 * on the file specified by the pathname. The method then writes the
 * argument anObject to the stream as binary data.
 * The argument implements the java.io.Serializable interface.
 */
public static void saveObject(Object anObject)
```

Write the code for this method. Your code must make use of the `File`, `ObjectOutputStream`, `BufferedOutputStream` and `FileOutputStream` classes. To achieve *full* marks your solution should inform the user of any exceptions that might occur and include a `finally` statement to ensure that once opened the stream is closed. [13]

(iii) Consider the following method comment and header defined for the hypothetical class `ObjectIO`.

```
/**
 * Prompts the user for a pathname and then attempts to open a stream
 * on the file specified by the pathname. The method then attempts to
 * read a serialised object from the stream.
 * The method returns either the retrieved object or null if no
 * object could be read.
 */
public static Object retrieveObject()
```

We don't want you to write the code for the whole method, just the code fragments specified in parts (a) and (b) below:

(a) Assume that a `File` object has been created and assigned to a variable called `objectFile` and that a variable called `inStream` has already been declared. Write the code that will assign to `inStream` a buffered stream object which can be used to read a serialised object from the file specified by `objectFile`.

(b) Assume that a variable `anObject` of type `Object` has previously been declared in the method. Write the line of code necessary to retrieve an object from `inStream` and assign it to `anObject`. [4]

[END OF QUESTION PAPER]