OLLSCOIL NA hÉIREANN
THE NATIONAL UNIVERSITY OF IRELAND, CORK

COLÁISTE NA hOLLSCOILE, CORCAIGH
UNIVERSITY COLLEGE, CORK

2016 – 2017

Semester 2 — Mock Exam 2017

CS2514 Introduction to Java

EXAMINERS

Dr Mock

DURATION

1.5 hours

INSTRUCTIONS

Answer **all 4** questions for full marks (80). There is **no need** to write whole classes *unless the question requires you write a class.* There is **no need** to write `import` statements. Use meaningful identifier names, pay attention to the layout, and make sure your coding style is clear. There is **no need** to provide comments, *unless the question requires you provide comments.*

SPECIAL INSTRUCTIONS

The use of calculators, books, notes, and crib sheets is *not* allowed.

PLEASE DO NOT TURN THIS PAGE UNTIL INSTRUCTED TO DO SO.
ENSURE THAT YOU HAVE THE CORRECT PAPER.

Question 1: Objects and Classes. *(20/80 marks)*

Question 1.a. *(4 marks)*
Using two sentences only, please define object state and object behaviour.
Question 1.b. *(6 marks)*
How do you implement object state and object behaviour?
Question 1.c. *(10 marks)*
Consider a class `Example` that defines a class attribute `attribute`, which is an `int`. Why can't this attribute represent object state?

Question 2: Inheritance. *(20/80 marks)*

Question 2.a. *(4 marks)*
Explain the notions of inheritance and polymorphism.
Question 2.b. *(4 marks)*
Explain the purpose of the '`IS-A` test'.
Question 2.c. *(8 marks)*
Explain the purpose of the '`@Override`' annotation. For sake of simplicity there is no need to mention abstract classes and interfaces in your answers.
Question 2.d. *(4 marks)*
Explain the notion of 'overloading' and provide a meaningful example.

Question 3: Class Design. *(20/80 marks)*

Using proper object-oriented class design, implement the classes in a class hierarchy for representing a `Book` in a `Book` shop. The following are the requirements.

- There are two kinds of `Book` categories: `PaperBook` and `EBook`.
- There are two `PaperBook` categories: `Hardcover` and `Paperback`.
- Each `Book` has a title, a page count, and a price.
- The price of an `EBook` is given by € 0.10 × p, where p is the number of pages of the `EBook`.
- The price of a `PaperBook` is given by € 0.15 × $p + c$, where p is the number of pages of the book and c the price of the book's cover.
- The cover price of a `Paperback` is given by € 1.00.
- The cover of a `Hardcover` book costs € 5.00.

To reduce the amount of work, there is no need to implement the class for `Hardcover` books. *However, you may not exploit this to simplify your class design.*

Question 4: Iterators and Generics. *(20/80 marks)*

Question 4.a. *(8 marks)*
Figure 1 depicts a contrived partial class that implements the `Iterable` interface. The only purpose of this contrived class is' to iterate over the members of the attribute `things`.

This class can be completed by providing code for the '`/* FILL IN #1 */`' and the '`/* FILL IN #2 */`' in Figure 1. Please complete the code for the two `FILL IN` parts. Make sure you clearly indicate the code for '`/* FILL IN #1 */`' and '`/* FILL IN #2 */`'.
Question 4.b. *(12 marks)*
Provide a generic class `Pair` for representing pairs of things. The class should provide a constructor, a method for getting the first member of the pair, a method for getting the second member of the pair, a method for setting the first member of the pair, and a method for setting the second member of the pair. The class should be parameterised over two types: one for the first member and one for the second member of the pair.

1 Answers to Questions

1.1 Semester 2 — Mock Exam 2017 (Answers)

Answer 1.a. Object state is what the object knows. Object behaviour is what the object does.
Answer 1.b. You define object state by defining instance attributes in the object's class and assigning values to these instance variables. You define object behaviour by defining instance methods in the object's class (and calling these methods).
Answer 1.c. A class may have more than one instance. Each instance may have a different state, i.e. each instance may have a different value for its instance attribute. The class attribute is shared among the instances of the class, i.e. the class attribute can only represent one `int` value. In general it's impossible for one `int` value to represent several `int` values.
Answer 2.a. In `Java` subclasses may extend a unique superclass. When a subclass extends its superclass, the subclass inherits all `public` attributes and methods from the superclass. This means the subclass can access any inherited attribbute and has the (default) method behaviour when calling an inherited method. The term polymorphism refers to the ability of a subclass instance to appear as an instance of its superclass.
Answer 2.b. The 'IS-A test' is used to decide if a given class can extend (be a subclass of) another given class. Specifically, if *A* and *B* are classes and if '*A* IS-A *B*' makes "sense" then *A* extends *B*.
Answer 2.c. A class which extends another class may override the inherited methods from the superclass. Overriding a method is done by providing a new definition for the method. The '`@Override`' annotation is meant to make the overriding more explicit and helps detect/prevent errors such as typos in the name of the overridden method.
Answer 2.d. If two methods from the same class have the same name, the same visibility, but different types for their parameter list then the methods are said to *overload* each other. The following two methods overload each other.

```
public void f( int a ) { }
public void f( int a, int b ) { }
```

Answer 3. The following qualifies for full marks.

```
public abstract class Book {
    private final String title;
    private final int pages;

    public Book( String title, int pages ) {
        this.title = title;
        this.pages = pages;
    }

    public final double getPages( ) { return pages; }
    public final double getTitle( ) { return title; }
    public abstract double getPrice( );
}

public class EBook extends Book {
    private static final double PRICE_PER_PAGE = 0.10;

    public EBook( String title, int pages ) {
        super( title, pages );
    }

    @Override
    public double price( ) {
        return PRICE_PER_PAGE * getPages( );
    }
}

public abstract class PaperBook extends Book {
    private static final double PRICE_PER_PAGE = 0.10;
```

```
    public PaperBook( String title, int pages ) {
        super( title, pages );
    }

    public abstract double coverPrice( );

    @Override
    public double price( ) {
        return PRICE_PER_PAGE * getPages( ) + coverPrice( );
    }
}

public final class Paperback extends Paperbook {
    private static final double COVER_PRICE = 1.00;
    public Paperback( String title, int pages ) {
        super( title, pages );
    }

    @Override
    public double coverPrice( ) {
        return COVER_PRICE;
    }
}
```

Answer 4.a.

- FILL IN #1:

```
implements Iterable<String>
```

- FILL IN #2:

```
public Iterator<String> iterator( ) {
    return new Iterator<String>( ) {
        private int index = 0;
        @Override
        public boolean hasNext( ) {
            return index < members.length;
        }
        @Override
        public boolean next( ) {
            return members[ index ++ ];
        }
        @Override
        public void remove( ) { }
    };
}
```

There's no need to override `remove( )`.

Answer 4.b. The following is a proper implementation.

```
public class Pair<S,T> {
    private S first;
    private T second;

    public Pair( S s, T t ) {
        first = s;
        second = t;
    }

    public void setFirst( S s ) {
        first = s;
    }

    public void setSecond( T t ) {
        second = t;

    public S getFirst( ) {
        return first;
    }
```

```
public T getSecond( T t ) {
    return second;
}
```

```
public class PartialIterableClass /* FILL IN #1 */ {
    private String[] things;

    public PartialIterableClass( String[] things ) {
        this.things = things;
    }

    /* FILL IN #2 */
}
```

Figure 1: Contrived partial class.