s

SCHOOL OF COMPUTING, ENGINEERING AND MATHEMATICS

**SEMESTER TWO 48 HOUR COURSEWORK**

**2019/2020**

**ANSWER SHEET**

**CI346**

**Programming languages, concurrency and client-server applications**

Student Number: 17806554

**Enter your answers to two questions from the exam paper below:**

1a)

**Statically typed:**

Statically typed languages allocate variables to a “type” as soon as they are declared, where a type refers to if a variable that is declared is a string, int, bool; i.e. data type.

*Pros:*

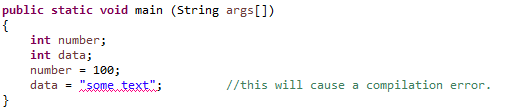
Due to variables being declared before runtime, this results in less runtime errors which makes the compilers job easier as errors are caught before the program is executed. Moreover, this means the script will not run till all errors have been fixed; making the debugging process more streamlined in some respects. Moreover, with static languages you get the assistance of an IDE for debugging leaving no ambiguity.

*Cons*

The need for declaration of types can lead to lengthy and overused type declarations which can result in complex error messages to require extra time to solve.

*Code snippet:*

//Java code snippet.



**Dynamically typed:**

A dynamically typed language means variables are only known and bound once the code has finished compiling.

*Pros:*

Dynamic languages are typically more concise than static languages resulting in less necessary code before anything useful can be made. Moreover, this results in important when it comes to server applications and when latency becomes important.

*Cons:*

Dynamic type languages are considered less secure as the data type is not checked till run-time. Moreover, this can be due to scripts in dynamic languages can be compiled whilst containing errors which could prevent the script from being run properly or at all.

*Code snippet:*

//JavaScript example



1b)

**L2 vs L1:**

L2 is safer than L1 as it is able to add protection to its abstractions and itself. For example, in language L2 an array can be used to store data creating a sequence of memory in a specific location. Moreover, the data stored in this array will be available when needed; however, a script written in L1 has development issues with bugs the content stored within the array may corrupt code written past it resulting in the corruption of other data structures. Furthermore, in L1 the lower level details would need to be much clearer to be able to determine where the issue lies, whereas, in L2 anything past the array will cause issues. In addition, parametric polymorphism (generic) means that code will work across multiple classes. For instance, with inheritance it enables the use of multiple classes and attributes where a method without polymorphism this would be so overly complex and demanding the system would be better off without it.

**L3 vs L2:**

If a function can be written that does not care what the input type is and doesn’t require knowledge at any level about this, parametric polymorphism is the one to use and can be written in one generic implementation of any type (L2). However, ad-hoc polymorphism will be needed otherwise which requires the implementation of each type (L3).

***Code snippets:***

distanceTravelled :: [ x ] -> Int

// shorthand for:

distanceTravelled :: forall x. [ x ] -> Int

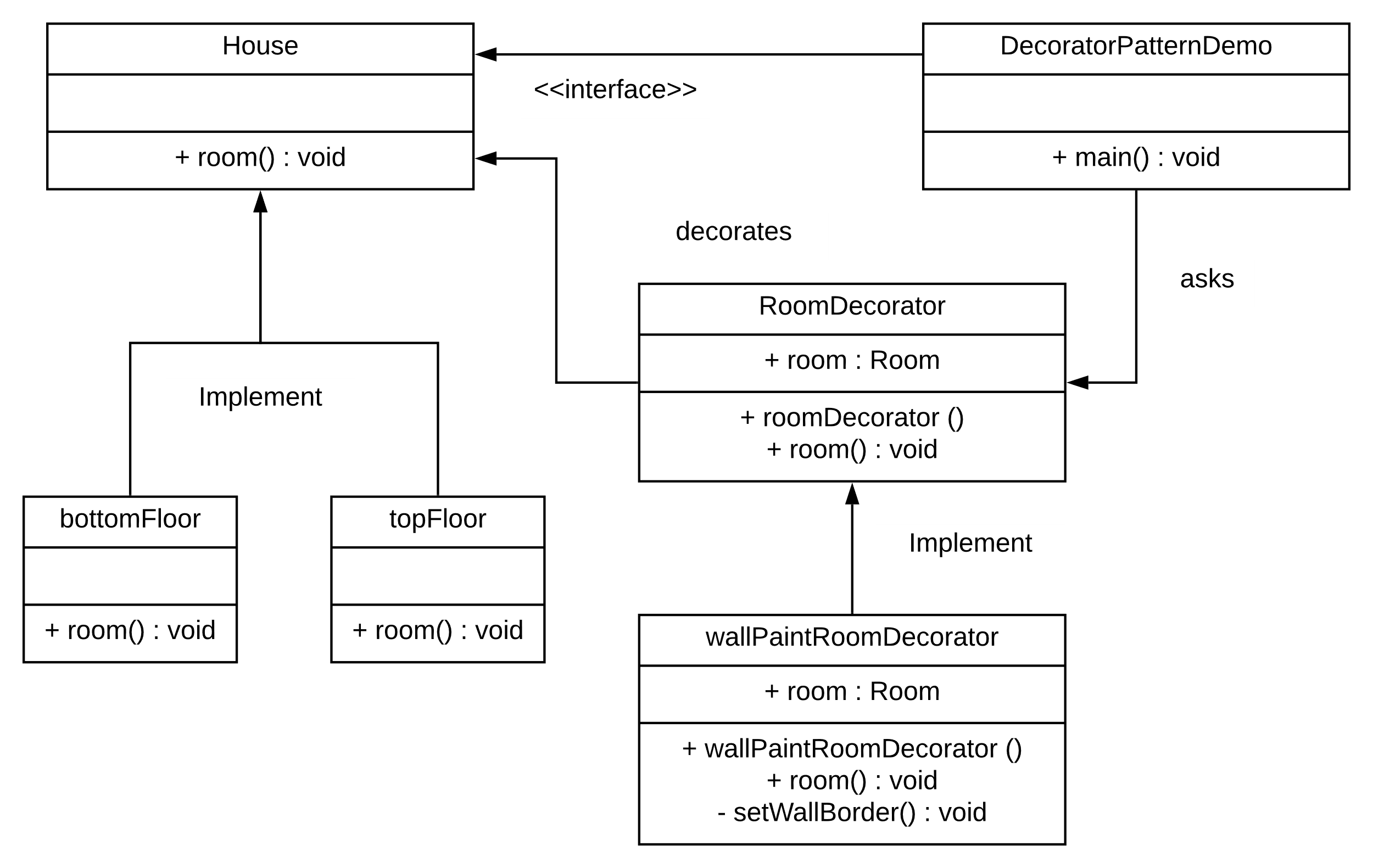
2a)

**Decorator Pattern:**

In an object-oriented language a decorator pattern can be used to add a set behaviour(s) to a object in a script in such a way that it does not affect other objects that are encompassed within the same class. The action of adding functionality to objects that are static or doing so via wrapping the other objects within the function at the runtime of the script by wrapping.

The first step creates an interface (house), next create a *concrete* class that implements the same interface (bottomFloor & topFloor). Once these interfaces are set up, we create a decorator abstract class that implement the roomDecorator interface and then extend the roomDecorator. The wallPaintRoomDecorator to decorate shape objects.

*Decorator UML:*



2b)

Functional programming at its core is declarative whereas OOP is mainly aimed at imperative programming. With the imperative approach the programmer will write some code that will describe with precise detail of the necessary steps taken by the computer to achieve a certain goal. However, in functional programming requires comprising a problem broken into sets of functions that will be executed. This requires the careful definition of each individual function and what it returns.

An imperative programming paradigm means a computer program describing the step sequence that change the state of a computer. However, a OOP paradigm are based on objects as a concept which can contain data classed as attributes. In contrast, functional programming builds elements and structures for computer programs that computes evaluations in mathematical functions that avoids changing-state and mutable data.

Imperative uses assignments directly and implements common data structures with global variables. In contrast, OOP bases its traits in objects and methods to pass messages and information and can achieve various tasks with inheritance and encapsulation whereas it’s counterparts cannot. Finally uses lambda calculus and recursion to achieve tasks, as well as, formulas and referential transparency.

# References

Anon., 2020. *Wiki.Haskell.* [Online]   
Available at: https://wiki.haskell.org/Polymorphism

Anon., 2020. *Wikipedia.* [Online]   
Available at: https://en.wikipedia.org/wiki/Comparison\_of\_programming\_paradigms  
[Accessed 2020].

Anon., n.d. *Tutorials Point.* [Online]   
Available at: https://www.tutorialspoint.com/design\_pattern/decorator\_pattern.htm

Burton, J., 2020. *Student Central.* [Online]   
Available at: https://studentcentral.brighton.ac.uk/webapps/blackboard/content/listContent.jsp?course\_id=\_113367\_1&content\_id=\_3385156\_1&mode=reset

Mairaw, N. N. G. N. M., 2015. *Microsoft docs.* [Online]   
Available at: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/functional-programming-vs-imperative-programming#:~:text=Imperative%20Programming,-The%20functional%20programming&text=Functional%20programming%20is%20a%20form,support%20impera

Zerexim, 2017. *Reddit.* [Online]   
Available at: https://www.reddit.com/r/haskell/comments/61mh3p/generics\_vs\_polymorphism/  
[Accessed 2020].