Executive Summary

What is the rationale for the use of object oriented programming and which limitations are overcome by use of functions in Java?

Object-oriented programming is the key programming methodology used today [1]. Examples of commonly used object-oriented programming languages include Java, C++, C#, Python and JavaScript. Some of the most important concepts in object-oriented programming include objects, classes, methods, inheritance, polymorphism and encapsulation. An object contains data in fields, which are variables of various types, including primitive variables such as integers, characters and boolean values as well as reference variables, which are references to other objects, such String, Person or Dog [2]. Objects are instances of classes [2]. A class contains the blueprint for a particular object [1]. Classes contain methods, which contain program statements that perform specific tasks [3]. An advantage of creating a method in a class is that it provides a way of performing a task that can be reused any time the task has to be performed, without rewriting the code.

The idea of inheritance comes from a hierarchical structure of classes inheriting features from other classes, where the subclass (or descendant) inherits from the super class (or ancestor) [3]. A subclass can override some methods contained in the super class and a subclass can only inherit from a single super class [4]. The concept of inheritance is important in programming as it allows code to be reused, saving time and resulting in less code to maintain.

Polymorphism is closely related to the idea of inheritance. It is a concept in object-oriented programming which enables the coding of programs with objects that extend a shared super class [1]. Polymorphism allows the call of the same method to result in different outputs depending on the object [1]. The main advantage of using polymorphism is that it simplifies code.

Encapsulation is another important component of object oriented programming. It involves the wrapping of variables and methods into a single unit [5] Encapsulation allows the interaction of objects while simultaneously keeping information about the implementation of an object (the variables and methods within the object) hidden [5].

Java is a programming language which heavily relies on the use of objects. A limitation of object-oriented programming in Java is that methods cannot be passed as arguments in other methods. A solution to this issue is the use of functions. Functions are pass as arguments in Java methods with the use of Lambda expressions. Other uses of lambda expressions in Java include:

- expression of one method interfaces in a concise way [6]
- improving performance and reducing the amount of code [6]
- allows the creation of functions outside of classes [6]

To summarise, object-oriented programming is used due to that fact that it minimises the amount of code that has to be written and maintained. The blueprint-like nature of object oriented programming makes it easier for software development projects to be undertaken. Java uses functions

to overcome the limitation of not being able to use methods as arguments and allows code to be more efficient.

References

[1] Paul J Deitel and Harvey Deitel. *Java How to Program, Early Objects*. Prentice Hall, 2014.

[2] Kathy Sierra and Bert Bates. *Head First Java: A Brain-Friendly Guide*. " O'Reilly Media, Inc.", 2005.

[3] C Thomas Wu. *An Introduction to object-oriented programming with Java TM*. McGraw-Hill Incorporated, 2006.

[4] Gopal Goyal and Sachin Patel. Importance of inheritance and interface in oop paradigm measure through coupling metrics. *Int. J. Appl. Inf*, 4:14–20, 2012.

[5] David Clarke. Object ownership and containment. 2001.

[6] Davood Mazinanian, Ameya Ketkar, Nikolaos Tsantalis, and Danny Dig. Understanding the use of lambda expressions in java. *Proceedings of the ACM on Programming Languages*, 1(OOPSLA):85, 2017.