**FIT2099: Exam Excerpts**

**Index**

Contents

[Keeping implementation details of classes private 1](#_Toc10990015)

[Risks of repeated code 1](#_Toc10990016)

[Fixing repeated code 2](#_Toc10990017)

[JavaDocs for private attributes 2](#_Toc10990018)

[Declared Exceptions vs Undeclared Exceptions 2](#_Toc10990019)

[Writing Executable Preconditions with Assertions and Exceptions 3](#_Toc10990020)

[Why not to cope with violated preconditions 3](#_Toc10990021)

[Access control for methods in preconditions 3](#_Toc10990022)

[Benefits of Writing Preconditions before Implementation 3](#_Toc10990023)

[Benefits of Writing Postconditions Before Implementation 3](#_Toc10990024)

[Command-Query Separation, Relation to Design by Contract 3](#_Toc10990025)

[Liskov Substitution Principle and Pre/Post Conditions 3](#_Toc10990026)

[Interface in Java 4](#_Toc10990027)

[Abstract Class in Java 4](#_Toc10990028)

[Interfaces vs Abstract Classes 4](#_Toc10990029)

[Why Reduce Dependencies 4](#_Toc10990030)

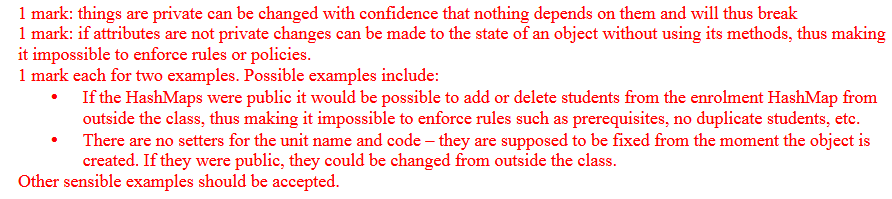
[Encapsulation and Access Modifiers to control dependencies 4](#_Toc10990031)

[What is good abstraction? Abstraction vs Abstract Classes 4](#_Toc10990032)

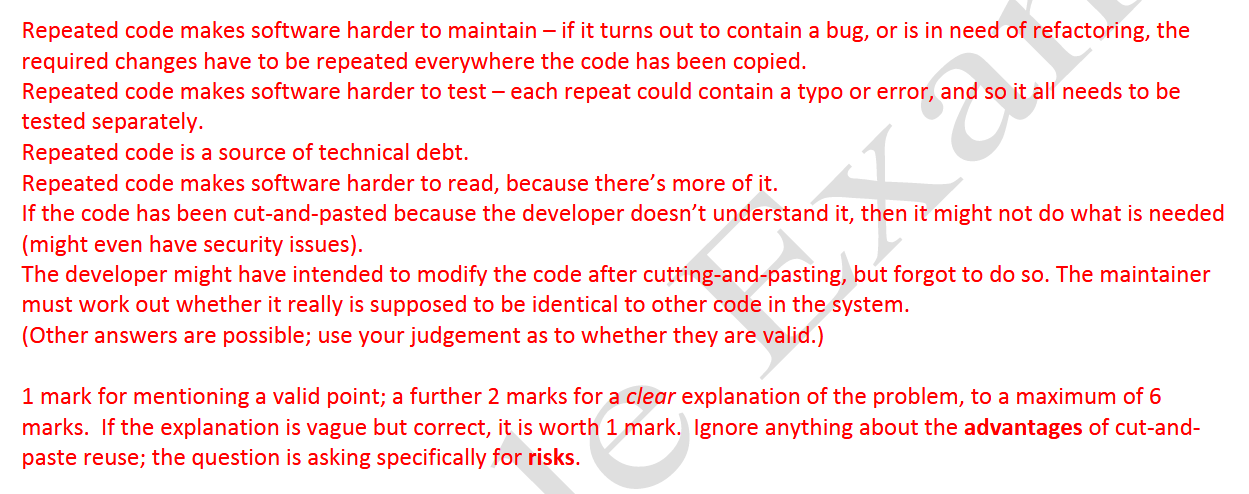
[More Specific Stuff 4](#_Toc10990033)

[Class Diagram and Sequence Diagram Mark Schemes 4](#_Toc10990034)

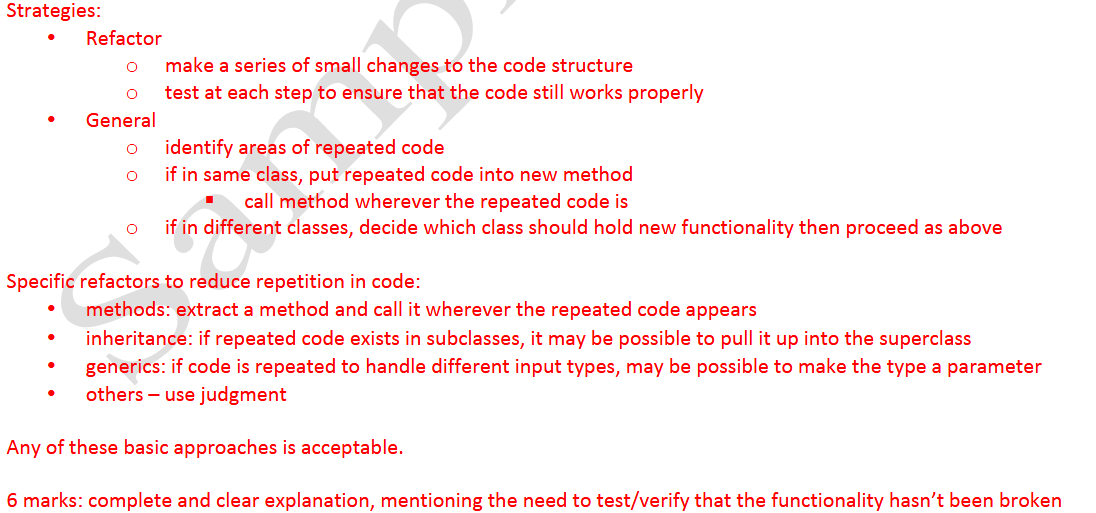
# Keeping implementation details of classes private

****

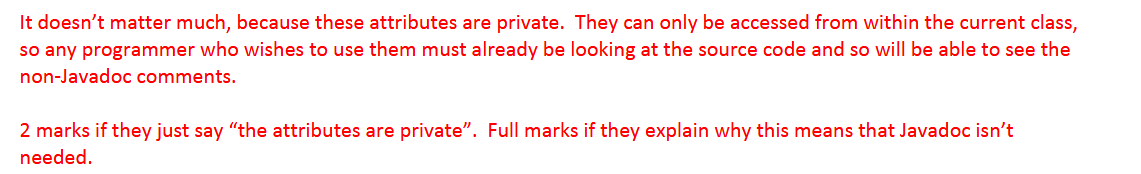
# Risks of repeated code



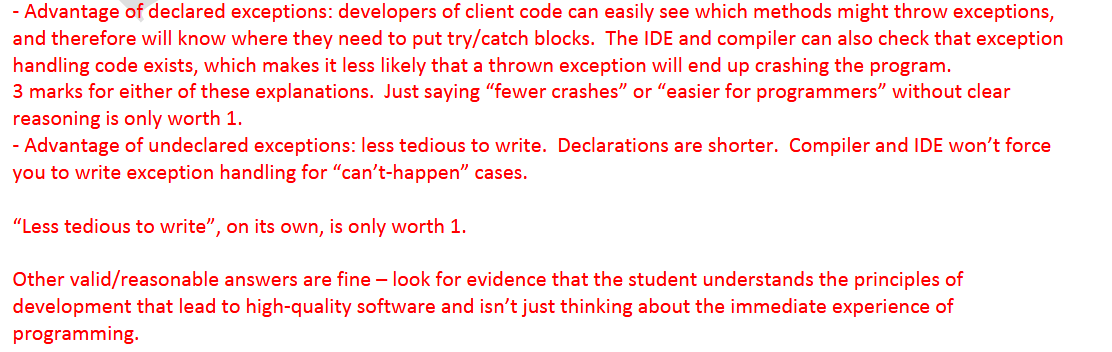
# Fixing repeated code



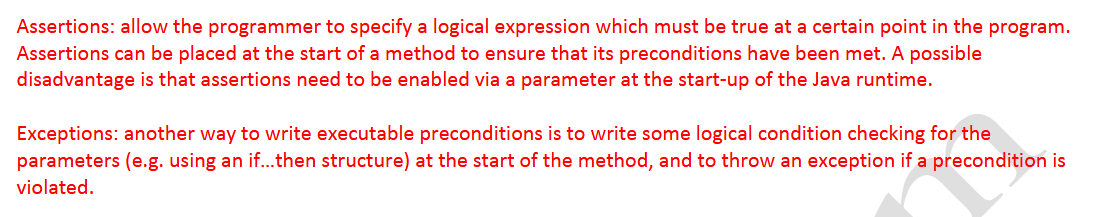
# JavaDocs for private attributes

****

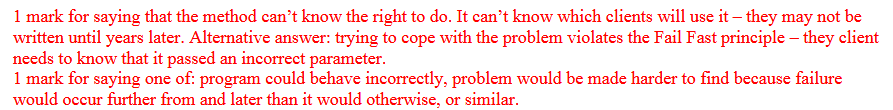
# Declared Exceptions vs Undeclared Exceptions

****

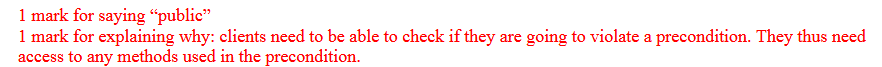
# Writing Executable Preconditions with Assertions and Exceptions

****

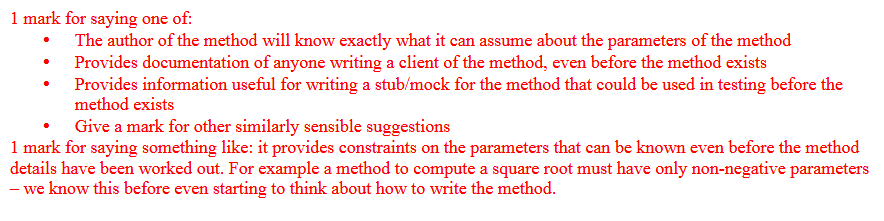
# Why not to cope with violated preconditions



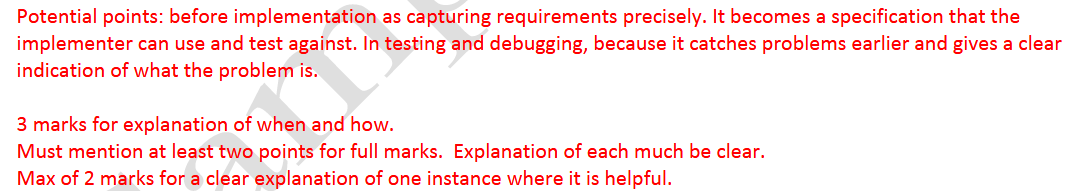
# Access control for methods in preconditions

****

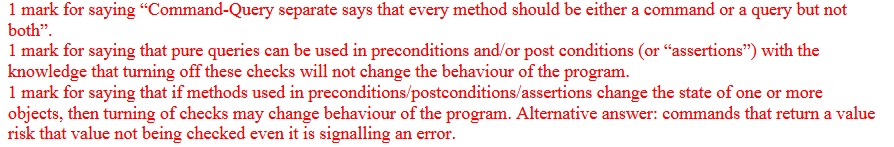
# Benefits of Writing Preconditions before Implementation

****

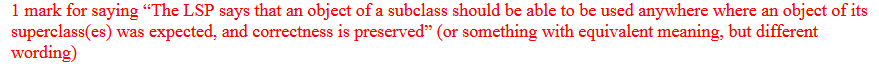
# Benefits of Writing Postconditions Before Implementation

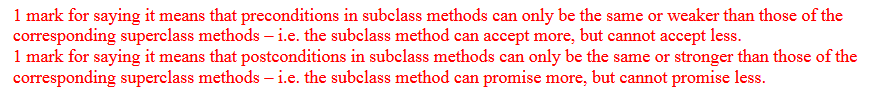
****

# Command-Query Separation, Relation to Design by Contract

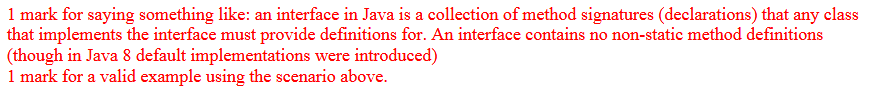


# Liskov Substitution Principle and Pre/Post Conditions

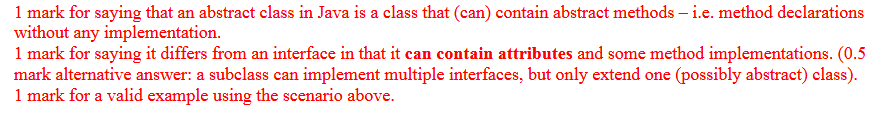
****

****

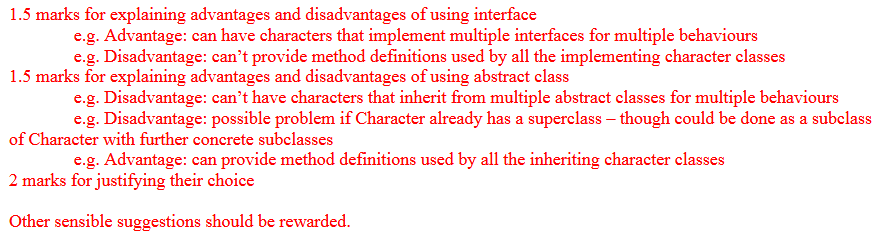
# Interface in Java

****

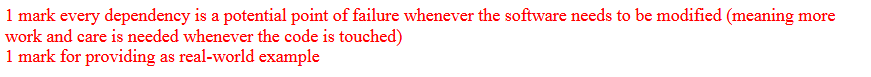
# Abstract Class in Java

****

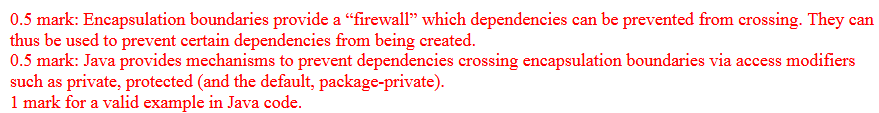
# Interfaces vs Abstract Classes

****

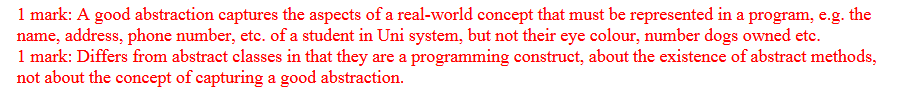
# Why Reduce Dependencies

****

# Encapsulation and Access Modifiers to control dependencies

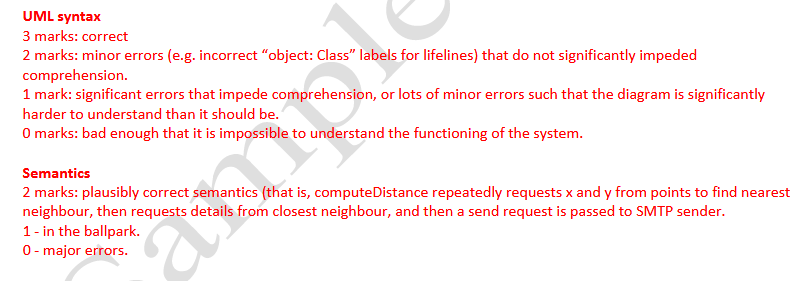
****

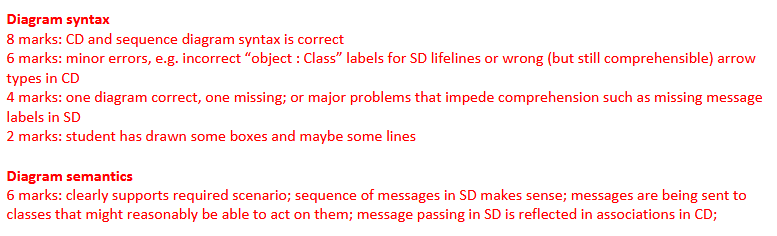
# What is good abstraction? Abstraction vs Abstract Classes

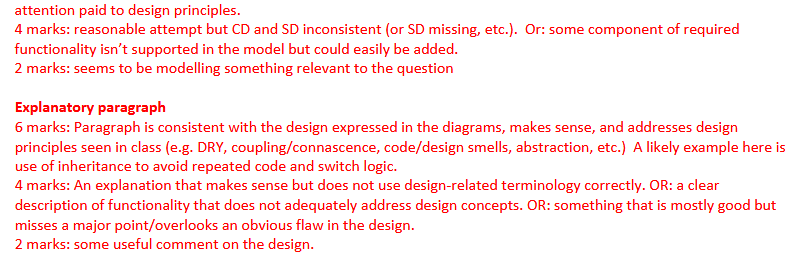
****

# More Specific Stuff

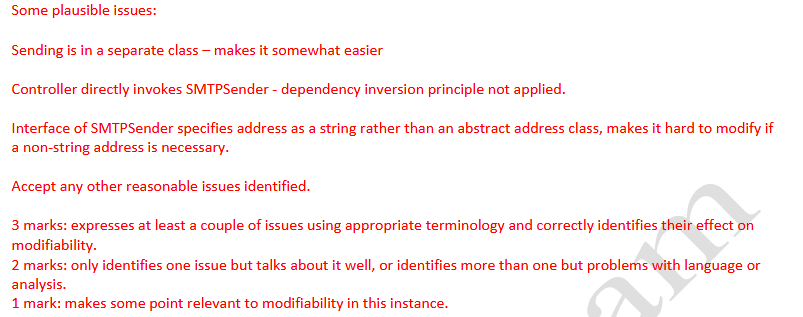
# Class Diagram and Sequence Diagram Mark Schemes



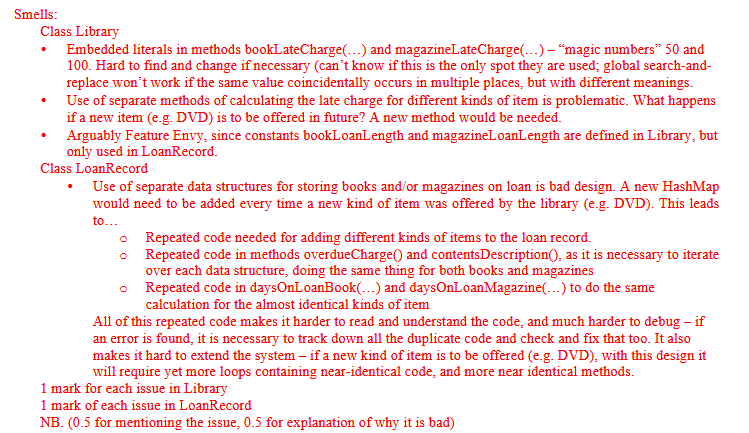




**Ease of Extending a System**

****

**Code Smells**

****