**FIT2099: Exam Excerpts**

**Index**

Contents

[Keeping implementation details of classes private 1](#_Toc10908367)

[Risks of repeated code 1](#_Toc10908368)

[Fixing repeated code 2](#_Toc10908369)

[JavaDocs for private attributes 2](#_Toc10908370)

[Declared Exceptions vs Undeclared Exceptions 2](#_Toc10908371)

[Writing Executable Preconditions 3](#_Toc10908372)

[Why not to cope with violated preconditions 3](#_Toc10908373)

[Access control for methods in preconditions 3](#_Toc10908374)

[Benefits of Writing Preconditions before Implementation 3](#_Toc10908375)

[Benefits of Writing Postconditions Before Implementation 3](#_Toc10908376)

[Command-Query Separation, Relation to Design by Contract 3](#_Toc10908377)

[Liskov Substitution Principle and Pre/Post Conditions 3](#_Toc10908378)

[Interface in Java 4](#_Toc10908379)

[Abstract Class in Java 4](#_Toc10908380)

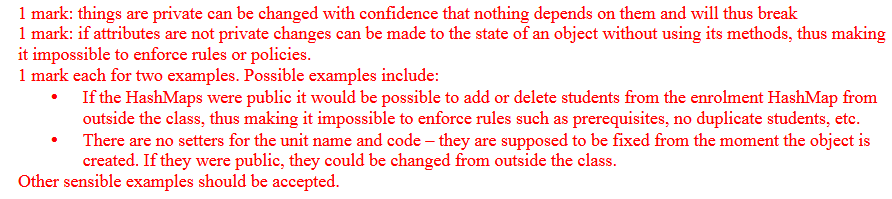
[Interfaces vs Abstract Classes 4](#_Toc10908381)

[Why Reduce Dependencies 4](#_Toc10908382)

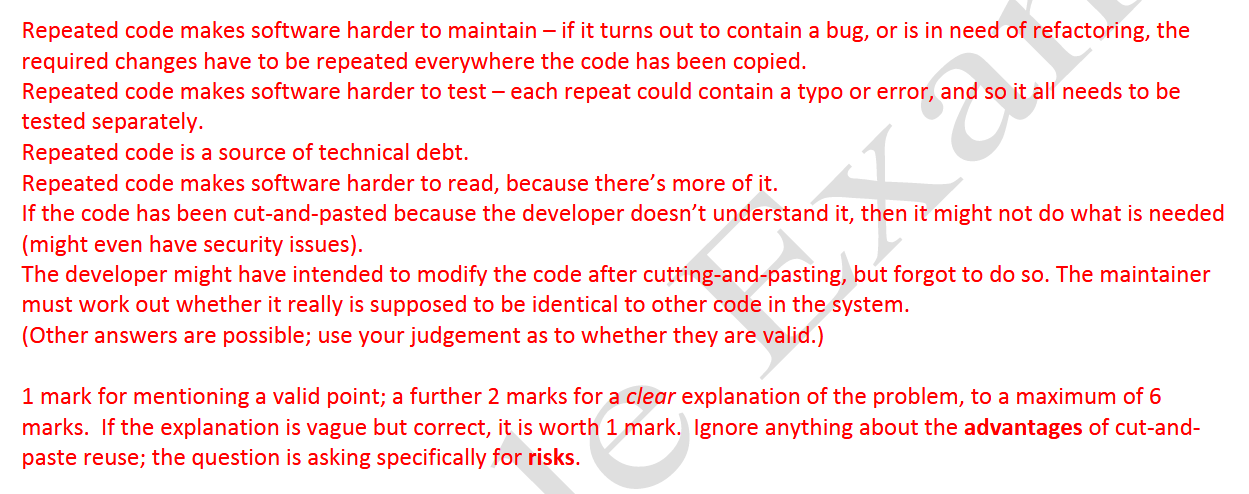
[Encapsulation and Access Modifiers to control dependencies 4](#_Toc10908383)

[What is good abstraction? Abstraction vs Abstract Classes 4](#_Toc10908384)

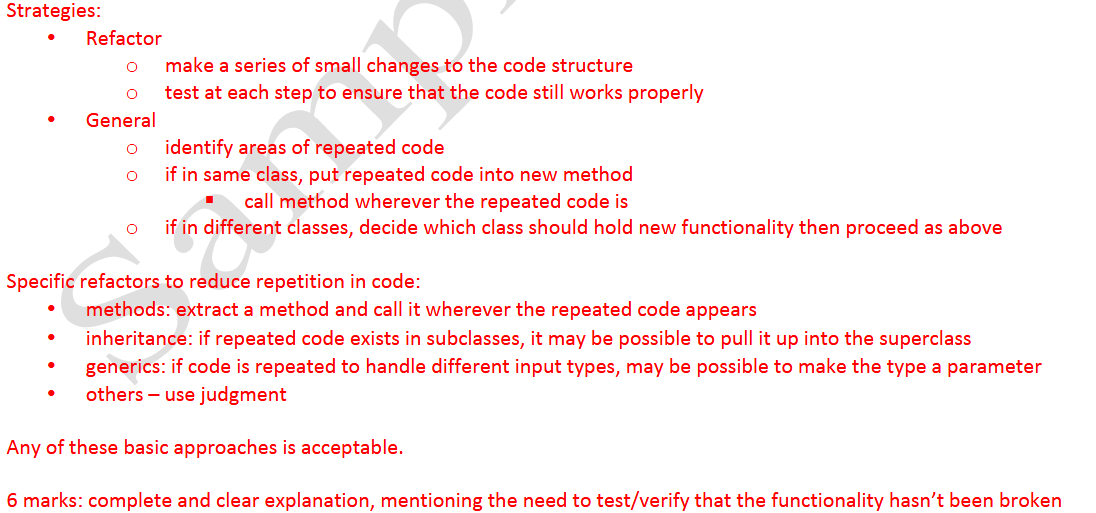
# Keeping implementation details of classes private

****

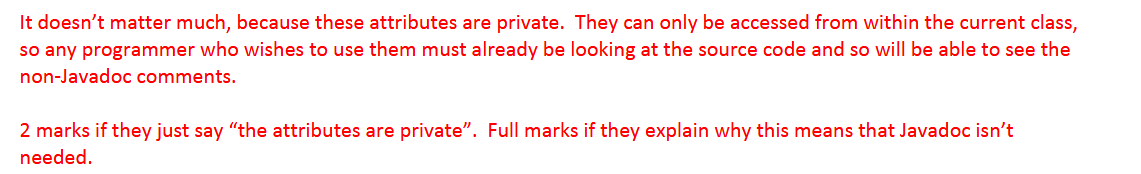
# Risks of repeated code



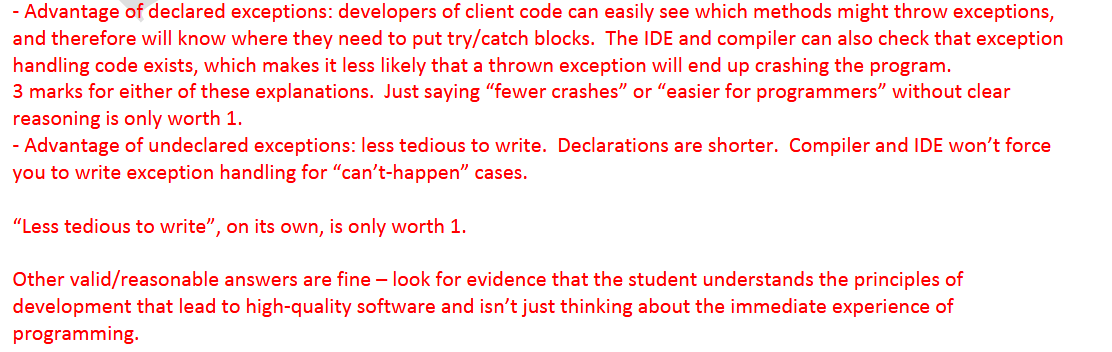
# Fixing repeated code



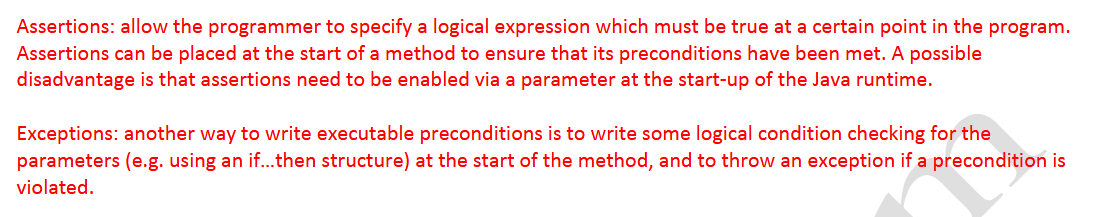
# JavaDocs for private attributes

****

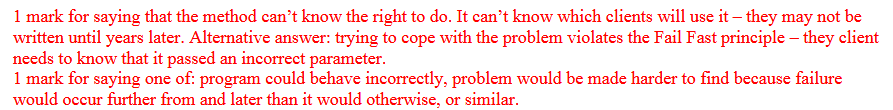
# Declared Exceptions vs Undeclared Exceptions

****

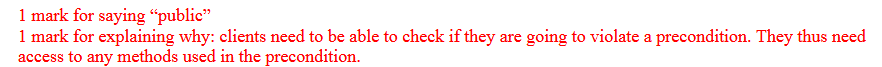
# Writing Executable Preconditions

****

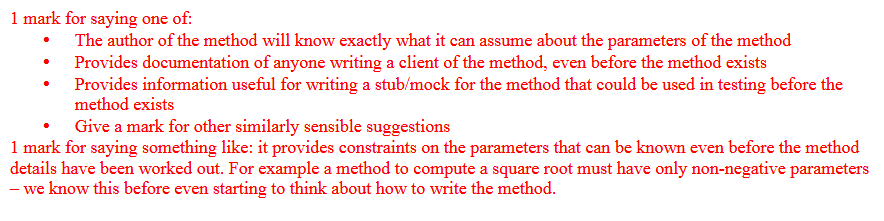
# Why not to cope with violated preconditions



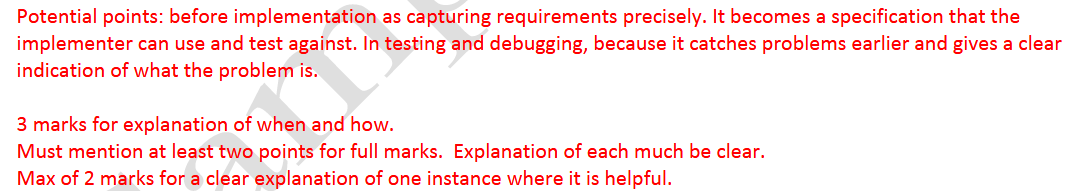
# Access control for methods in preconditions

****

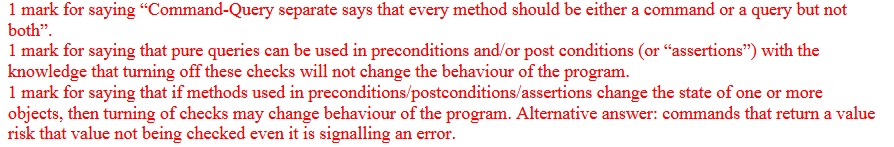
# Benefits of Writing Preconditions before Implementation

****

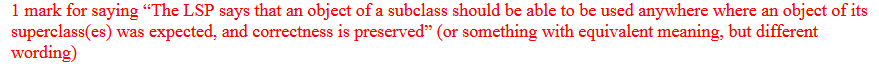
# Benefits of Writing Postconditions Before Implementation

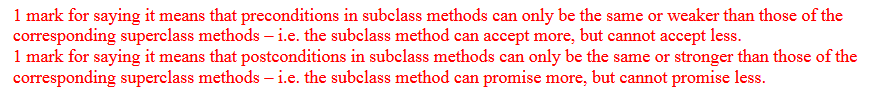
****

# Command-Query Separation, Relation to Design by Contract

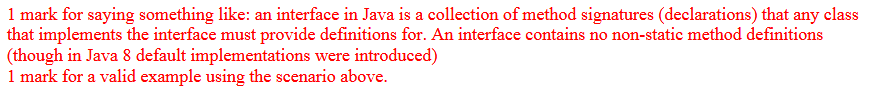


# Liskov Substitution Principle and Pre/Post Conditions

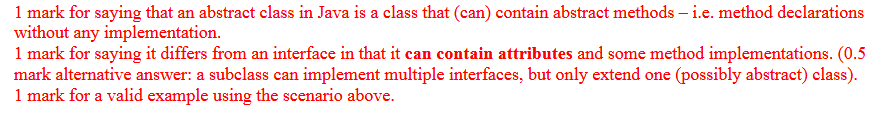
****

****

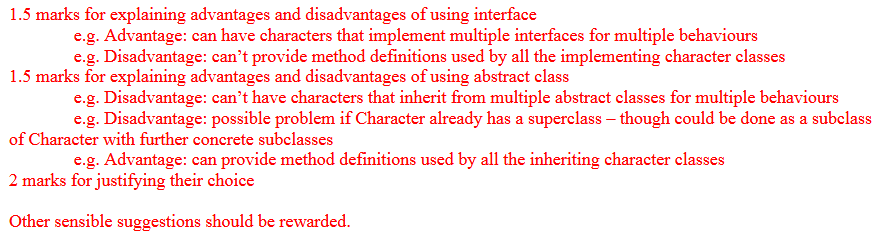
# Interface in Java

****

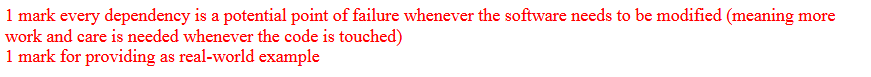
# Abstract Class in Java

****

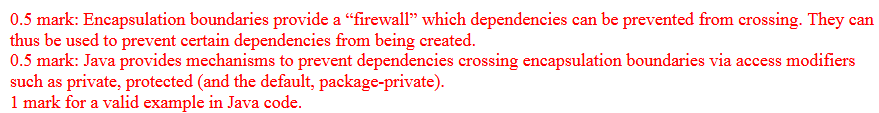
# Interfaces vs Abstract Classes

****

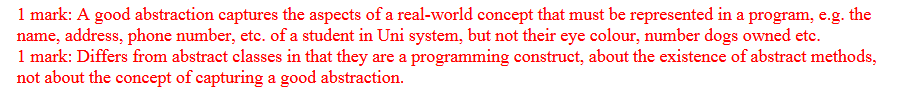
# Why Reduce Dependencies

****

# Encapsulation and Access Modifiers to control dependencies

****

# What is good abstraction? Abstraction vs Abstract Classes

****