Manus för muntlig tentamen

TDA552 - Objektorienterad Programmering och Design

Alex Gerdes, Niklas Broberg, Jonas Amlström Duregård

Hösttermin 2018

1 Allmän information

Frågorna under muntan kommer att komma från manuset nedan. Alla kommer inte få samma frågor, men räkna med att ni får åtminstone någon fråga per lärandemål. Ytterligare frågor kan förekomma, i form av följdfrågor utifrån era svar på frågorna nedan.
Frågor markerade (X) är designade för att fiska efter svar som visar på högre kvaliteter (4 eller 5). Detta betyder inte att den som kan svara på dessa frågor automatiskt får 4/5, eller vice versa. Det går att besvara även andra frågor på ett sätt som visar på att ni uppnått en högre kunskapsnivå; det går likaledes att besvara (X)-frågor på ett sätt som visserligen är korrekt, men som inte visar på högre kunskapsnivå.

För att få betyget 4:a krävs det att man visar ett högre kunskapsnivå i *två* lärandemål och för betyget 5:a för alla lärandemål. Lärandemålet 'Grundläggande objektorienterande koncept' är ett undantag, den räknas inte med för högre betyg.

2 Manus

Frågorna är uppdelade per lärandemål. För godkänt betyg behöver ni kunna svara på grundläggande frågor om vart och ett av lärandemålen.

2.1 Redogöra för objektorienterade design-principer

För några av SOLID-principerna, samt för någon av de övriga principerna vi tagit upp under kursen:

1. Vad säger principen?
2. Ge något konkret exempel på hur man bryter mot, eller följer, principen.
3. Vad fyller principen för syfte? Varför är det bra att följa principen? (X)

SOLID-principerna är:

Single Responsibility Principle	Open-Closed Principle
(X) Liskov Substitution Principle	Interface Segregation Principle
Dependency Inversion Principle	

De andra principerna är:

Separation of Concern	Law of Demeter
High Cohesion, Low Coupling	Command-Query Separation Principle

2.2 Designmönster, samt deras syfte och effekt

1. Vad är ett design pattern?
2. Varför använder man design patterns? (X)

För några av följande design patterns:

Template Method	Strategy	State
Bridge	Decorator	Adapter
Factory	Observer	Singleton
Chain of Responsibility	Iterator	Composite
Module	Facade	MVC

3. Vad är pattern P för något? När och hur kan man använda det?
4. Vilka designproblem löser man genom att använda pattern P? (X)

2.3 Grundläggande objektorienterade koncept

Redogör för begreppen eller begreppsparen[1]:

Klass och objekt	Statisk och dynamisk typ	Inkapsling
Implementationsarv	Specifikationsarv	Subklasser
Statisk och icke-statisk metod	Dynamisk bindning	Konstruktor
Variabel och attribut	Primitiv typ	Referenstyp, alias
Abstrakt klass, interface	Overloading, overriding	

2.4 Avancerade språkmekanismer och tekniker

Redogör för:

Mutability och immutability	Defensive copying	Method cascading
Refactoring	Exceptions	Mutate-by-copy
Trådar, trådsäkerhet		

[1] Dynamisk bindning används här i betydelsen *dynamic dispatch*. Termen dynamisk bindning har även andra betydelser som ligger utanför kursinnehållet.

2.5 Arv, parameteriserade typer, code reuse, polymorfism

Frågor om centrala begrepp:

1. Vad är polymorfism? Vad är subtypspolymorfism? Vad är parametrisk polymorfism?
2. På vilket sätt blir kod bättre av att man använder parametrisk resp. subtypspolymorfism? (X)
3. Vad är kovarians? Vad är kontravarians?
4. Vad är delegering? Vad är arv?
5. Hur åstadkommer man kodåteranvändning med delegering resp. arv?
6. När bör man använda arv? Varför säger vi att man ska föredra komposition framför arv (composition over inheritance)? (X)

Frågor av mer teknisk eller Java-specifik karaktär:

7. Vad är en typkonstruktor? Vad är en typparameter?
8. Vad är en typvariabel? Vad är ett wildcard?
9. I vilken utsträckning tillåter Java varians vid metod-overriding?
10. Vad innebär nyckelorden extends och super?
11. När bör man använda extends och super? (X)