Spring Professional Exam Tutorial v5.0

Question 01

Question 01 - What is the difference between checked and unchecked exceptions? Why does Spring prefer unchecked exceptions? What is the data access exception hierarchy?

Checked exception – Exception that is extending `java.lang.Exception` (expect `java.lang.RuntimeException`) class that has to be explicitly declared in `throws` part of method signature of method that is throwing an exception and has to be explicitly handled by code that invokes the method. If code that is calling the method with checked exception does not handle exception, it has to declare it in throws part of method signature.

Pros:

- Developer using API always has a list of exceptional situations that has to be handled
- Fast compile-time feedback on check if all exceptional situations were handled

Cons:

- May result in cluttered code
- Coupling between callee and caller

Question 01 - What is the difference between checked and unchecked exceptions? Why does Spring prefer unchecked exceptions? What is the data access exception hierarchy?

Unchecked exception – Exception that is extending `java.lang.RuntimeException` class, does not have to be explicitly declared in `throws` part of method signature of method that is throwing an exception and does not have to be explicitly handled by code that invokes the method. Developer has freedom of choice if error handling should be implemented or not.

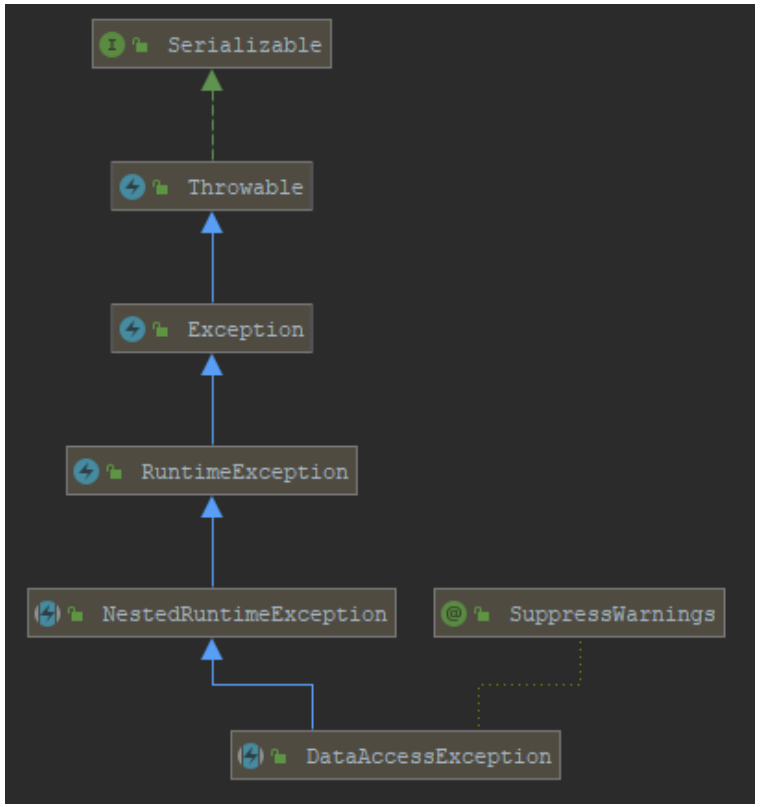
Pros:

- Reduces cluttered code
- Reduces coupling between callee and caller

Cons:

- May result in missing situations in which error handling should be implemented
- Lack of compile-time feedback on error handling

Question 01 - What is the difference between checked and unchecked exceptions? Why does Spring prefer unchecked exceptions? What is the data access exception hierarchy?



- Data Access Exception is a Runtime Exception
- Examples of concrete Data Access Exceptions
  - CannotAcquireLockException
  - CannotCreateRecordException
  - DataIntegrityViolationException
- Purpose of this hierarchy is to create abstraction layer on top of Data Access APIs to avoid coupling with concrete implementation of Data Access APIs