2nd scenario of comparison – "Which class is more cohesive?"

Main_Config2 vs. DB_Helpers:

- **With orange background, explanation from the 10 responses which chose the second class (DB_Helpers) as more cohesive**
- **With yellow background, 15 explanation from the responses which chose both classes having quite similar cohesion**
- **With blue background, 54 explanation from the responses which chose the first class (Main_Config2) as more cohesive**

13	No primeiro código é possível carregar e salvar as configurações e a classe está limitada a isso. Já o segundo código possui diversas funções de diferente contextos como por exemplo, identificar o próximo ID disponível na tabela, deletar uma informação, preencher um combo box, etc.
15	**Ainda sim, não acho que a classe DB_Helpers possui coesão suficiente. Em alguns métodos usa String pra referenciar a tabela e em outros métodos utiliza um DefaultModel. A classe Main_Config2 está longe de ser uma classe com coesão. Possui informações do hotel e ainda faz o controle das reservas de forma procedural.**
16	Main_Config2 apresenta um menor número de funcionalidades e dependências em relação à DB_Helpers.
17	Eu achei a primeira mais coesa uma vez que faz a tarefa específica de configuração. A segunda tb é específica de banco, mas realiza operações de consulta e deleção. Creio que poderia ser mais coesa.
18	Porquê a classe Main_Config2 é mais específica... A segunda classe pode ser dividida em mais de uma para que fique bem específica, da maneira atual tem métodos de objetivos diferentes.
19	A classe DB_Helpers implementa muitos comportamentos que podem ser entendidos como modulos separados
20	A primeira classe apenas realiza a leitura do arquivo e possui os métodos de get e set nas propriedades de configuração do sistema, enquanto a classe 2 possui apenas métodos genericos relacionados a banco de dados.
22	se considerarmos um contexto mais abrangente (funções diversas de gerenciamento de dados no banco) temos algum grau de coesão, embora não haja coesão funcional. ambas as classes se enquadram neste contexto mais abrangente.
24	A segunda classe possui mais funções.

25	Novamente, esses helpers são difíceis de definir sua funcionalidade e fazem suposições a respeito dos módulos que fazem uso da mesma (formato da query, modelo do banco, etc.). Sem falar que realiza coisas que não são de sua responsabilidade, como tratar exceção do banco, se é uma classe "helper" ela não tem informações suficientes pra entender o que aconteceu... por exemplo, quem disse que o erro deve gerar uma mensagem e não é um erro "esperado"?
26	Both classes lack cohesion. DB_Helpers is both specific and generic. Having the method getNextID which is generic while is specific to query specific tables in DB. It should be separated. The Main_Config should not load and store the infos form config file. It should be done by 2 different classes. One to store info and a second one to load it from the config file.
28	Nem olhei direito a segunda, mas ela tem misturado componentes gráficos e acesso a banco de dados. Isto é, tem que se preocupar com muita coisa.. cria uma acoplamento que não deveria existir.
29	Maybe I am doing something wrong. But it seems to me like the previous case. Cohesion in MainConfig2 due to presence of shared attributes and all methods in DB_Helpers are static.
30	Ambas possuem baixo nível de coesão. Para obter uma maior coesão, a primeira Main_Config2 deveria ser dividida em duas classes uma para os gets e outra para o sets. A segunda, DB_Helpers possue diversas funcionalidades que deveriam ser distribuídas em mais classes. Por exemplo, "Returns the number of rooms in the database" deveria estar em uma classe separada. O mesmo para "Returns the number rooms in use at the specified date d" e outros métodos.
32	As classes tem propósitos diferentes o que a meu faz com que a comparação entre elas não seja direta, mas as duas parecem semelhantes respeitando seus objetivos.
33	A primeira classe contém vários métodos que ocupam diversos atributos e outros métodos da própria classe. No caso da segunda classe, isto não ocorre e esta poderia ser decomposta em mais classes.
34	**A primeira classe possui rotinas que não são exclusivamente ligadas a carregar configurações de um arquivo. A classe acabou absorvendo informações de acesso ao banco de dados.**
35	A classe DB_Helpers foi alimentada com a velha preguiça de fazer um modelo legal para resolver o problema. Pode ter ate começado com boas intenções (talvez métodos como getNextID) mas depois começou a adicionar diversas funcionalidades que poderiam estar mais bem separadas (exemplo: getNumberOfRooms) Cade a classe Hotel por exemplo? Que teria o numero de quartos disponiveis em vez de colocar em um Helper.
36	A primeira tem uma responsabilidade mais definida. Asegunda é muito genérica e trata de muitas responsabilidades.

37	Estou considerando que a classe de Helpers possui métodos variados, que não correspondem à uma funcionalidade coesa, e sim várias funcionalidades diferentes. Já MainConfig possui a funcionalidade de gerenciar arquivos de configuração, sendo na minha opinião, mais coesa.
38	Ambas são classes utilitárias, com variedade de métodos para servir outras classes.
39	A primeira classe só executa funções ligadas a ler e carregar um arquivo de configuração, porém a segunda realiza diversas funções diferentes, apesar de serem com o mesmo fim, mas que vão de encontro ao de que uma boa coesão se dá quando é assumida somente uma responsabilidade
41	A classe Main_Config2 tem o papel bem definido de gerenciar as propriedades de configuração do sistema. Já a classe DB_Helpers, apesar de ter diversos métodos relacionados ao banco de dados, são métodos que não tem o mesmo sentido, por isto esta classe não tem uma função bem definida no sistema, o que a torna uma classe com baixa coesão.
42	**a classe DB Helpers faz apenas o que se propões que é métodos para acesso a banco de dados mantendo sua coesao, a classa Main_Config2, além de obter dados de configuração do sistema, busca informações que nao tem essa finalidade, tais como: endereco do hotel,fax, telefone etc.**
43	A segunda classe é responsável por diversos métodos com diferentes funções, assim a classe perde a coesão.
44	**MainConfig2 is grouping the application proprieties, while DB_Helpers is mixing database operations with interface.**
46	A primeira esta bem definido a sua função de carregar as configurações, enquanto a segunda deixa "aberto" que possui diversas finalidades, podendo realizar diversas tarefas distintas.
50	A classe DB_Helpers mistura conceitos, possui métodos que são especificos para a entidade "Room" quando deveria possuir somente métodos genericos para ajudar nas operações com banco de dados. Isso pode se tornar um problema na manutenção se existe alguma outra classe que se encarrega de salvar, alterar e apagar dados com relação a entidade "Room". Já a classe Main_Config2 de fato se encarrega somente em carregar e prover acesso as configurações gerais do software.
51	Uma vez que a classe DB_Helpers apresenta métodos com nomenclaturas específicas de outro domínio que não tem a ver com comportamentos de um banco da dados.
54	A classe DB_Helpers é muito dependente de elementos da interface gráfica (Swing). Além de controlar o BD, ela insere elementos em listas e combos da interface Java. Isso implica em baixa coesão da classe.

56	The first class is solely related to configuration management (retrieving options from a config file, and saving back any changes). The second class seems a real mess: although the name of the class is DB_Helpers, most of the services in it do not seem to me mere helpers; for example, there are methods for checking if rooms can be deleted which are more related to the business rules of the system (rooms cannot be deleted if they are booked), than to the manipulation that should be done in the DB in order to get the work done.
57	DB_Helpers possui métodos relacionados a outras coisas além de prover métodos diversos de acesso ao banco de dados.
58	O raciocínio é similar ao da questão anterior. A segunda classe acaba desempenhando muitas funções dentro de um mesmo contexto o que poderia ser dividido em novas classes ao passo que a primeira cuida de apenas uma operação.
59	Segui o mesmo raciocínio que segui na questão anterior.
60	A classe DB_Helpers se parece com uma classe utilitária, no entanto além de oferecer métodos comuns às operações de banco, ela também possui algumas regras de negócios relacionadas às reservas e aos quartos do hotel.
61	As duas classes não são muito coesas, pois realizam várias operações referente a diferentes conceitos.
62	Nenhuma das classes me pareceram suficientemente coesas por misturarem elementos de acesso ao Banco de Dados (e.g., porta, host, password), controle de reservas (e.g., invoice) e controle de quartos (e.g., numero de quartos disponíveis)
63	DB_Helpers possui métodos sem relação um com o outro, como método para calcular próximo ID de uma tabela e métodos para consultar informações sobre quartos por exemplo.
64	The DB_Helpers class deals with different concerns, as it mixes generic methods to handle the database with methods to check whether a room is deletable, for instance. On the other hand, Main_Config2 is only concerned with loading, saving and managing the configuration properties.
65	Eu achei as duas classes com pouca coesão. A primeira, Main_Config2, que deveria focar só em ler e salvar as configurações de um arquivo, expõe mais comportamento através dos métodos públicos get e set para cada atributo. Dessa forma um cliente pode usar essa classe para outras funções além de ler ou salvar as configuração de um arquivo. Pode definir as configurações diretamente com os métodos gets e sets. A segunda classe possui vários helpers que só tem em comum a relação com o banco de dados. Mas alguns helpers são gerais, como o delEntry, outro são específicos de uma tabela, como o isRoomDeletable. Acredito que essa classe poderia ser dividida melhor. Talvez uma classe responsável pela tabela com informações dos quartos. Outra classe utilitária mais geral.
66	**Achei que as responsabilidades da segunda classe são mais bem definidas do que na primeira**

67	**Executa diversas responsabilidade, como: - Buscar o próximo id livre de uma tabela; - Deletar um registro de uma tabela; - Procurar por quartos livres em uma data; etc**
69	Novamente, segui a heurística de ler a responsabilidade de cada classe e verificar se os métodos públicos de cada classe estavam ou não relacionados com a responsabilidade da classe. Desta forma, identifiquei que para a classe DB_Helpers há uma série de métodos que realizam responsabilidades bastante distintas: alguns atuam diretamente no banco, outros em elementos da interface, por exemplo. Por isso considerei DB_Helpers menos coesa. Além disso, eu empreguei novamente o conceito de que classes altamente coesas são indivisíveis. Desta forma, julguei que DB_Helpers poderia ser quebrada em uma classe contendo métodos auxiliares que atuam na camada de acesso ao banco de dados, e outra classe contendo métodos auxiliares que atuam na camada de apresentação. obs.: Não considerei os métodos getters e setters na análise de coesão.
70	The amount of different accessor methods (getters/setters) in the first class led me to consider this as a low cohesive class. Those methods deal with several purposes. It is better to have several getter/setter classes, that join similar responsbilities, than having a "global" one, that joins accessors to all entities in the project. The same problem is faced in the second class. It doesn't have a single and well-focused objective.
71	A segunda classe possui acoplamento com diversas outras classes. Dessa forma, a coesão da classe diminui já que existem vários outros elementos que não fazem parte da implementação da funcionalidade principal da classe
72	pelos mesmos motivos apresentados anteriormente
73	A primeira classe tem como responsabilidade fazer o load e o save de configurações. Enquanto a segunda classe mistura um pouco as responsabilidades, por exemplo, os métodos isRoomDeletable e getComboItems não parecem estar realizando atividades de um mesmo contexto.
74	The fields from the first class are similiar regarding the behavior of this class (more similar than those from the second class).
75	Os atributos e métodos das duas classes não tem tanta relação uns com os outros
77	Apesar de apresentarem a mesma "dificuldade" em fragmentação em uma possível refatoração, a classe Main_Config2 parece ter mais sentido por concentrar as funcionalidades que concentra, principalmente pelo conjunto de gets e sets
79	A segunda classe executa procedimentos que não deveriam ser feitos, por exemplo queries para insert.
83	Não sei o suficiente sobre coesão para conseguir perceber qual tem o maior nível de coesão :(
84	A forma como a primeira classe foi mais estruturada que a segunda, deixando a classe mais simples de entender.

86	A principio a classe DB_Helpers é uma classe que contém os métodos que não encontraram seu devido lugar no sistema :) Consequentemente, a coesão fica prejudicada.
88	As duas classes são bastante coesas e têm poucas dependências para outras classes do software, sendo dificilmente quebradas em classes menores (o que geraria muitas dependências entre as novas classes formadas)
89	A segunda classe mistura métodos de delete da BD com métodos de busca para preencher combo box, gerenciamento de ocupação dos quartos etc sendo assim pouco coesa, enquanto a primeira realiza apenas a configuração do aquivo.
90	As duas classes tem o mesmo nível de flexibilidade. A diferença é que em uma você tem a robustem de um SGBD para garantir as propriedades ACID e na outra você teria a "leveza" de um arquivo XML. Nas duas situações (Banco e arquivo XML) você poderia acesso a informação de maneira estruturada (no banco pelo paradigma relaciona e pela XML usando as tags).
91	A classe MainConfig2 é altamente coesa uma vez que seus elementos são altamente relacionados internamente, principalmente nos métodos load_config e save_config. Já a classe DB_Helpers apresenta uma baixa coesão. Além de seus elementos não estarem fortemente relacionados internamente, ela classe pode ser facilmente decomposta em outras classes.
93	DB_Helpers has operations not only related to the DB, but also related to the business logic (related to a rooms availability) whereas the Main_Config2 has operations related to the Config only.
94	O objetivo da primeira classe é bem definido, o da segunda não é. A segunda classe mistura regras de negócio específicas da reserva de quarto, ao mesmo tempo que fornece um método que delete qualquer tipo d entrada no banco de dados. A mesma poderia ser dividida em duas ou mais classes com objetivos mais específicos.
95	A classe DB_Helpers mistura responsabilidades ao manipular elementos como combo box.
97	**Although the Main_Config2 is acting like a config file but it is better that for different related classes, we use different config class. This class is not cohesive, it contains all information to config DB, Invoice, Billing, storing guest data and so on. So it does not focus on specific functionality. Helper Class helps to handle the functionalities of DB so it is more cohesive than the Main_config one.**
98	Same as my previous answer: Main_Config2's members heavily use other members of the same class. That's not really the case for DB_Helpers.
99	**The Main_Config is tangled with fields for the hotel as well as for the database. It could easily be separated in two util classes.**

102	**Both classes are pretty terrible for cohesion. DB_Helpers is a utility class that mixes UI code with model code. It is really quite awful, but it is all relevant to the DB. Main_Config2 is a smaller class, but its role is 2 fold, it contains DB information and billing information and currency information. I claim it is less cohesive because the distance from UI to Data is shorter than the distance of DB Config to Customer information. Both are bad.**
105	I look at the behavior. The first class has varied data elements, but its behavior can be summarized as load config data, save config data, and retrieve any particular config data item. The second class has many different behaviors, at multiple layers of abstraction, from db level concerns (getNextId, delEntry) to UI concerns (getComboItems) to business level operations (getNumberRooms).
106	Very little can be said without further context. These both seem like utility classes, and they do have clearly defined functionality.
107	The first class, Main_Config2, its responsibility was clear to me; a instance of this class holds configuration settings for the system and provides methods for serializing/deserializing those information to/from an XML file. I prefer to split the responsibility to following three parts: 1) holding hotel information, 2) holding database information, 3) serialization and deserialization. As for the second class, DB_helpers, it is again a static utility class and its responsibility is ambiguous to me. Some concerns are overlapped; The fist three methods, getNextID and two delEntry, are utilities to wrap SQL queries in a table-independent way. Following two methods, getComboItems and getListItems, populates items into GUI models by querying the database. Remaining three methods, isRoomDeletable, getReservedRoomsAtDate and getNumberOfRooms encapsulates queries to specific tables in the database.
110	O Main_Config2 está muito duas responsabilidades: - Ler/Escrever arquivo de configuração - Fazendo papel de Entity O DB_Helpers está com muitas responsabilidades, até mesmo para uma classe Helper.
113	These 2 classes implement “miscellaneous” functionality. The first one (Main_Config2) deals with all the configuration settings (language, hotel information, database, billing policies, etc.) and the second one (DB_Helpers) deals with various operations on the database (delete entries, get number of rooms, etc.). These classes are not as cohesive as the previous 2 classes (DB_Backend and DB_InsertUpdate). However, in my opinion Main_Config2 is slightly more cohesive than DB_Helpers because DB_Helpers implements functionality from the solution domain (deleting entries in the table) as well as problem domain (get number of rooms), whereas Main_Config2 only deals with the settings (however diverse they are).
114	**The first one Main_Config2 merge both hotels data and database data The second one have only information to help the sue of database**
115	The methods in Main_Config2 provide access to a program configuration, though the actual data members may be not related to each other. Some methods in DB_Helpers are not related to each other, and they are located in DB_Helpers because they are just helper methods.

116	O cenário é parecido com o anterior. O agravante da segunda classe é que ela disponibiliza métodos que não parecem ter relações com um único conceito, ou seja, métodos não relacionados. Por exemplo, essa mesma classe lida com conceito de ID, com o gerenciamento de quartos etc Por outro lado, é simples dizer o que a primeira classe faz - lida com a configuração do sistema. Perceba que os métodos dessa classe acessam/modificam quase que unicamente os atributos dessa própria classe.
117	A primeira classe possui um papel bem definido, enquanto a segunda realiza diversas operações de banco, tanto de persistência quanto de consulta, além de realizar consultas a tabelas específicas, quando sua função proposta era de ser uma classe utilitária. Os métodos de consulta a tabelas específicas deveriam estar em classes criadas especificamente para acesso a estas tabelas.
118	Considering there is a class dedicated to only updating the database, the same can be done to delete or get information from the database. In other words, BD_Helpers can be separated in a class for only queries and another for only deleting. However, I don't understand why a configuration needs all that info, why database info must be with name, phone, etc.
119	The first class (Main_Config2) can be separated in two classes (one to set, another to get). The other class looks like is just a class that has been used to put methods related to a DB that do not fit anywhere else.
122	Again, the first class is more cohesive. It provides access to the system configuration, serving the role of a kind of "Configuration Manager". The private fields are reused in different methods and the common responsibility of the class is very clearly cut. The second class is just a random collection of helpful methods, that mostly do not have any contextual similarities in their responsibilities. Their code is probably much better suited to be put into classes that are responsible for the respective tasks, e.g. a "DeleteEntryCommand".
123	Na primeira clase Main_config2 eu nao consegui identificar o uso de outras classes Na segunda classe DB_Helpers eu percebi que ela usa a classe DB_Backend
124	Na minha opinião a primeira classe é coesa pois se restringe apenas a carregar parâmetros gerais de configuração do software, e manipular esses parâmetros internamente (métodos get e set). Por outro lado, a segunda classe perde coesão a partir do momento em que funções que lêem informações do banco de dados se misturam com funções que manipulam os formulários utilizados na interface do usuário.
125	apesar da classe Main_Config2 apresentar 2 operações (save_config() e load_config()) distintas do seu propósito inicial de setar e recupear parâmetros de configuração a classe DB_Helpers apresenta operações de contextos muito distintos de um do outro (get_nextID(), del_entry(), getListItems(), etc.). em outras palavras, a primeira diz respeito somete à configuração enquanto a segunda contém operações genéricas em tabelas bem como relacionadas a booking ou ainda a rooms.