3rd scenario of comparison – "Which class is more cohesive?"

RelationSpouse vs. RelationParentChild:

- **With orange background, explanation from the 6 responses which chose the first class (RelationParentSpouse) as more cohesive**
- **With yellow background, explanation from the 30 responses which chose both classes having quite similar cohesion**
- **With blue background, explanation from the 43 responses which chose the second class (RelationParentChild) as more cohesive**

13	Neste caso as duas classes são classes filhas da superclasse Relationship. Na primeira estão todas as funções para o relacionamento de marido e esposa. Na segunda estão todas as funções para o relacionamento de pais e filhos.
15	A classe RelationSpouse contém um método printToFile que não tem nada a ver com o propósito da sua própria classe.
16	As classes possuem funcionalidades e dependências similares.
17	**Tem um método na RelationSpouse (printToFile) que poderia ser extraído para uma classe específica pra ficar mais coesa**
18	São bem objetivas e tem um comportamento específico.
19	O comportamento print to file poderia ser facilmente implementado em um Visitor especifico para impressão em arquivo.
20	As classes foram construidas com o mesmo nível de coesão, pois todas as verificações relacionadas a pessoa são realizadas na classe especifica (Person)
22	Embora a impressão no arquivo para a classe RelationSpouse se refira aos seus dados, este método de impressão poderia ser delegado para outra classe (específica para impressão).
24	Classes com objetivos bem definidos.
25	**Não me parece apropriado colocar a responsabilidade de gravar em disco na classe RelationSpouse. Nesse caso ela possui mais de uma responsabilidade e por isso é menos coesa.**
26	RelationSpouse was suppose to be a relation only class but it includes a JPanel reference. Not cohesive at all.
28	Na minha opinião nenhuma é coesa, num ponto de vista de design. Isto é, a classe Pessoal deveria mantar estes relacionamentos. Além disto, mesmo motivo da questão anterior.. a primeira classe se preocupa com a responsabilidade dela e mostrar coisas na tela..

29	My decision is influenced by the method printToFile in RelationSpouse. I don't think this class should be responsible for writing data.
30	A classe RelationSpouse possui baixa coesão e deve ser divida em mais classes, como por exemplo o método "printToFile" deveria estar em outra classe.
32	Para mim as classes são tem objetivos similares, ambas implementações são parecidas. Isso faz com que a coesão seja parecidas também.
33	As duas classes são muito semelhantes, usando atributos e métodos internos. No entanto a classe RelationSpouse contém um método (ou um comportamento) que poderia, talvez, ser separado desta classe (método printToFile). A classe RelationParentChild não inclui este comportamento de imprimir para um arquivo.
34	A primeira classe além de representar o relacionamento entre marido e mulher possui métodos que não estão ligados a esta atividade, por exemplo, o método printToFile.
35	Ambas cuidam de relacionamento entre duas entidades e cada uma com seus próprios critérios.
36	A segunda é mais coesa, proque a prmeira trata da impressão em arquivo que não deveria ser de sua responsabilidade.
37	Elas possuem métodos similares/equivalentes, que gerenciam um determinado relacionamento entre parentes de uma árvore genealógica.
38	As duas classes tem a mesma lógica e implementa suas soluções da mesma maneira.
39	a primeira classe assumiu mais responsabilidades, como imprimir e tratar erros, além de apresentar a relação entre marido e mulher, por isso acredito que ela é menos coesa
41	A primeira classe além de representar o relacionamento entre marido e mulher também mostra os relacionamentos em uma interface gráfica, enquanto a segunda classe apenas representa o relacionamento entre pais e filhos.
42	as duas classes trata apenas do que se propõe.
43	Cada uma classe executa métodos específicos.
44	RelationSpouse mixes domain concepts with printing it to a file.
46	Ambas fazem a mesma tarefa que é de verificar o relacionamento entre duas pessoas.

50	A classe RelationSpouse possui método para a serialização em arquivo. Isso deve ser evitado pois a serialização deve ser algo ortogonal a classe que representa uma entidade, e independente de método (arquivo, banco, stream). Já a classe RelationParentChild se resume somente a representar esses dados e ter metódos para comparação (se é igual etc.) e manuseamento da entidade, o que é bastante consistente.
51	a classe RelationSpouse assume comportamentos não relacionada ao relacionamento entre marido e mulher, como por exemplo a responsabilidade de gerar um arquivo.
54	A classe RelationSpouse usa elementos da interface gráfica do Java, o que não deveria ocorrer para se alcançar uma alta coesão. O mais adequado seria lançar exceções e os componentes da interface gráfica fazerem o tratamento das mensagens.
56	The first class includes a method that does not have any relationship with the information the class must maintain in the system: printToFile.
57	RelationSpouse tem outras coisas além da relação marido e mulher como o método de imprimir.
58	No meu entendimento, as classes são similares. A única diferença é que atuam em domínios diferentes (homem-mulher e pais-filhos). Por essa razão, considero a coesão como sendo a mesma para elas.
59	Se olharmos ao propósito de casa uma das classes, então elas são igualmente coesas, pois o propósito de cada uma é bem claro e distinto. Se olharmos à estruturação das classes tendo em vista a reutilização de métodos dentro da própria classe, então elas são igualmente coesas, pois não considero que poderia ser alcançado um nível mais alto de reutilização dos métodos dentro da própria classe.
60	A classe RelationParentChild é coesa pois ela só atende aos interesses da relação entre pais e filhos. A classe RelationSpouse não apresenta coesão visto que além de tratar da relação do casal ela também exibe mensagens de interface gráfica e faz output para um arquivo.
61	RelationSpouse contém a operação de escrever em arquivo que poderia ser uma classe separada, se tornando mais coesa.
62	Ambas as classes realiza funções relativas à relação, e não à esposa ou ao filho especificamente, como pode parecer em um primeiro momento.
63	o método RelationSpouse.printToFile() poderia ser movido para outra classe especializada em tratar saída dos dados, deixando a classe RelationSpouse mais coesa.
64	I picked RelationParentChild only because RelationSpouse has an additional method for printing the relationship to a file at the end that did not need to be in the class.
65	Eu acho que a primeira classe RelationSpouse possui um método printToFile que deveria ser responsabilidade de outra classe, por isso, acho ela menos coesa que a segunda.

66	não consegui definir qual das duas classes foi mais coesa
67	A primeira tem uma responsabilidade de gravação em arquivo que não deveria ser atribuida à classe.
69	Novamente, comecei buscando as responsabilidades de cada classe, mas desta vez peguei as responsabilidades do próprio enunciado da questão, não dos comentários. Pela própria definição das responsabilidades, identifiquei que havia definição de mais de uma responsabilidade, o que me soou logo de cara como um sintoma de baixa coesão. Analisando o código fonte, percebi que a classe RelationParentChild implementa apenas a lógica relacionada ao relacionamento entre pessoas, enquanto a classe RelationSpouse implementa a lógica relacionada ao relacionamento entre pessoas e também as lógicas de salvar no arquivo e exibir na interface gráfica mensagens. Seguindo a sugestão inicial de que uma classe altamente coesa é indivisível, julguei que RelationParentChild é mais coesa do que RelationSpouse, visto que RelationSpouse poderia ser quebrada em três classes: uma para representar a lógica da relação entre pessoas, outra para a exibição de mensagens na interface gráfica e outra para a persistência em arquivos.
70	The first class has a method that doesn't have to do with its main purpose, but rather to "print" things, that indeed could serve to more than this single class and, as such, could be considered as a separate entity. This problem doesnt happen to the second class, that only involves things related to its main purpose.
71	Ambas as classes possuem o mesmo nível de coesão nesse caso apesar da forma de implementação/verificação das relações entre os objetos
72	acredito que o nível de dificuldade em dividar as duas classes é o mesmo
73	A classe RelationSpouse possui o método printToFile que não deveria fazer parte desta classe para que a coesão fosse mantida mais alta.
74	The fields from both classes are similiar regarding the behavior of each class.
75	As duas classes são iguais
77	A classe RelationSpouse tem como objetivo definir os atributos e métodos relacionados à husband e wife, com adição da funcionalidade de escrever em arquivo. A funcionalidade de I/O parece ser mais facilmente realocada em uma outra classe, se comparado a qualquer método da classe RelationParentChild
79	a primeira tem regra de negocio que nao deveria pertencer a ela
83	As duas são bem parecidas e todos os atributos estão lá possuem uma única responsabilidade (como definido no inicio da pesquisa)
84	Achei as classes muito parecidas.

86	A classe RelationSpouse utiliza os recursos de apresentação do Swing, portanto é menor coesa pois sua reutilização em outras tecnologias de apresentação ficaria prejudicada.
88	Ambas dependem das mesmas classes.
89	As duas classes fazem um tratamento similar as relações familiares, uma trata no nível marido/mulher e na outra filho <-> pai/mãe
90	**A classe RelationSpouse é mais coesa devido a necessidade de utilizar diversos imports. import java.io.File; import java.io.FileWriter; import java.io.IOException; import java.util.Iterator; import javax.swing.JOptionPane;**
91	Ambas as classes possuem o mesmo nível de coesão. Em ambos os casos o elementos relacionam-se internamente. Além disso, os atributos são utilizados pela maioria dos métodos implementados pela classe, o que indica uma forte coesão.
93	RelationSpouse has printing related functionality in it. It should be moved to a Printer class.
94	As classes possuem objetivos bem definidos.
95	A classe RelationSpouse tem responsabilidades que não são condizentes com seu contexto. ex: printToFile
97	Both handles specific functionality.
98	I don't see substantial differences in the usage pattern of members.
99	RelationSpouse also handles children, but children are not necessary for husband and wife; on the other hand, parents are necessary for children, so the second class is more cohesive
102	Similar classes but class 1: RelationSpouse couples itself with File IO directly. This is totally unnecessary, a toString() would be alright but FileIO is unweidly and complicated and you probably wanted that string anyways. FileIO is not relevant to a relationship. Immediate string serialization is.
105	The printToFile class in RelationSpouse does not belong. It is not behavior that is related to the relationship.
106	Now these, since they are very similar, I can make a judgment about. THe first class is more cohesive since the relationSpouse has a method related to printing its information to file, but relationParentChild does not. That being said, I do not necessarily thin that it is wrong to include the print method in relationParentChild. THough it is less cohesive, it might be a good decision to put the print method inside this class. More context is needed (i.e. are all other Relation classes similar to Spouse or to ParentChild). I favor consistency over any metric such as cohesion.

107	The second class contains a single responsibility to maintain parent-child relationship between domain objects. On the other hand, the first class, RelationSpouse, has extra responsibilities to report exceptions on a GUI widget and dump debug information in addition to the basic responsibility that maintains spouse relationship between domain objects. Comparing objects using reference equivalence (==) in RelationParentChild#isParent() and #isChild() considered to be buggy, although I know this is not relevant to the survey.
110	A RelationSpouse está com muita responsabilidade. Ela não deveria escrever em arquivo e também não deveria ser responsável pela View. Eu não modificaria o RelationParentChild.
113	These 2 classes are almost equally cohesive. However, the RelationParentChild class seems to be a bit more cohesive than the RelationSpouse because the RelationSpouse class has the method printToFile(File f) which deals with Input/Output functionality, and which is not coherent with the remaining fields and methods of the class that deal with “relation” issues.
114	**Bith classes have a problem with the method getPartner(). I think it is a method for Person. The class RelationParentChild is more cohesive because of the method printToFile() in the class RelationSpouse that make it less cohesive**
115	This is the difficult question, but printToFile method in RelationSpouse class may be located in another class that defines a format of textual output.
116	A segunda classe é mais coesa que a primeira. A primeira classe, além de lidar com seus atributos e possuir métodos que lidam com as relações marido-esposa, possui um método para imprimir em um arquivo os dados. Para mim, isso afeta a coesão, pois a classe lida com mais de um conceito (relação marido-esposa e persistencia de dados).
117	A segunda classe possui alta coesã, enquanto a primeira extrapola suas responsabilidades ao realizar envio de mensagens de alerta e gravação em arquivos.
118	**The RelationParentChild seems to be recycled from the other class. Why there is a methods called "getPartner", the intention is not clear. Besides, both class have methods called "getPerson1" and "getPerson2" which does not make a lot of sense.**
119	Given that both classes have similar functionality (to find out the relation between two relatives), and the methods are similar, I decided to determine that both classes have similar cohesion.
122	Here, both classes contain the code for a specific relation, which is correctly specified in both cases. The only difference is, that the first class is actually mixing other code into its behavior (like the JOptionPane calls), which has nothing to do with its intended responsibility (which I think should be just the representation of a relationship). Thus, the second class is more coherent, because it does not mix in different concerns like the display of messages.
123	Ambas são subclasses de Relation

124	Neste caso a segunda classe é mais coesa do que a primeira, pois enquanto que a primeira classe mistura métodos destinados a manipular arquivos em disco com métodos que tratam do relacionamento marido/mulher (responsabilidade principal da classe), a segunda trata APENAS do relacionamento pais/filhos, mantendo assim a sua coesão.
125	**ambas as classes apresentam métodos semelhantes, aparentemente herdados da classe Relation. no entanto a classe RelationSpouse tem um método que caiu do céu para salvar salvar em arquivo.**