Carimbo de data/hora	Pontuação	Name and Student ID	What is the name of the file you evaluated?	How would you rate your confidence that there is a REFUSED BEQUEST smell in the code?	Mention why (for instance, which code elements) you gave this rating.
27/04/2025 14:02:45		UFMG #15	IncompleteElementException.java	1	Classes de excessões geralmente não usadas de uma maneira a chamar o super() com argumentos. Neste caso temos a classe usando bem o comportamento esperado da classe pai BuilderException.
28/04/2025 22:38:06		UFMG #38	SaTokenException	2	It is an exception class. I think it redefines some methods, but I do not see this as refused bequest.
29/04/2025 22:20:30		UFMG #42	RandomGenerator	1	A classe sobrescreve verify mas, a superclasse AbstractGenerator provavelmente declara verify como abstrato. Isso é um uso legítimo de herança, não está recusando.
01/05/2025 18:59:35		UFMG #18	RelationshipTest	3	Fora o uso de super.before(), não aparenta haver nenhum uso das funcionalidades da classe pai extendida.
01/05/2025 19:58:25		UFMG #18	AckSchedulerKey	1	A classe aparenta ter métodos com responsabilidade bem definida e em linha com a classe extendida.
03/05/2025 14:11:13		UFMG #42	ConversionException	1	Não há rejeição de funcionalidades. Usa bem herança.
06/05/2025 07:47:38		UFMG #11	ConversionException	2	Acredito que não há code smells, a não ser que a classe pai faça algo diferente com a mensagem de exceção, mas acredito que não seja o caso.
06/05/2025 07:54:01		UFMG #11	ScriptProcessorBuilder	3	A classe parece depender de outras classes internas as bibliotecas declaradas.
06/05/2025 07:58:11		UFMG #11	CellFormulaTagHandler	2	Não identifiquei code smells.
06/05/2025 16:36:20		UFMG #20	redis.clients.jedis.exceptions	3	A classe apenas estende JedisDataException e não adiciona nenhum comportamento ou lógica adicional, apenas define construtores. Isso pode ser considerado uma classe desnecessária se não tiver um propósito claro de especialização.
14/05/2025 17:43:12		UFMG #17	ImmutableEnumMap	4	EnumSerializedForm herda métodos de Form mas os ignora, implementando os próprios métodos.
14/05/2025 17:43:19		UFMG #10	LexemePath	2	Creio que o código não apresenta de fato um refused bequest, já que ele não apresenta duas classes, sendo uma herdando de outra. Porém, ela definitivamente possui métodos longos, com vários blocos if aninhados em alguns métodos.
14/05/2025 17:48:45		UFMG #38	HazelcastPubSubStore	5	A classe CellFormulaTagHandler acessa e modifica diretamente o estado interno de XlsxReadSheetHolder em todos os métodos, o que quebra o princípio de encapsulamento. Isso indica que essa lógica provavelmente deveria estar dentro do próprio XlsxReadSheetHolder. Refatorar usando métodos como iniciarFormula(), adicionarFormula() e finalizarFormula() deixaria o código mais limpo, coeso e fácil de manter.
14/05/2025 17:48:53		UFMG #39	AbortedTransactionException	5	A classe AbortedTransactionException está extendendo da classe JedisDataException, porém não implementa nada de novo nos métodos, está somente repassando os mesmos parametros para a clase super, então não há sentido em ela existir.
14/05/2025 17:49:33		UFMG #4	IncompleteElementException.java	5	The class is literally calling the parent's methods and doing nothing else,other than callind an existing method.
14/05/2025 17:52:35		UFMG #17	PendingEntry	3	Talvez exista um code smell, talvez não. Não consegui identificar.
14/05/2025 17:52:59		UFMG #6	HelpCommand	1	Não há o smell especificado no código, apesar de haver outros smells.
14/05/2025 17:53:30		UFMG #16	CellFormulaTagHandler	5	A classe CellFormulaTagHandler acessa e modifica diretamente o estado interno de XlsxReadSheetHolder em todos os métodos, o que quebra o princípio de encapsulamento. Isso indica que essa lógica provavelmente deveria estar dentro do próprio XlsxReadSheetHolder. Refatorar usando métodos como iniciarFormula(), adicionarFormula() e finalizarFormula() deixaria o código mais limpo, coeso e fácil de manter.
14/05/2025 17:55:19		UFMG #17	btActivatingCollisionAlgorithm	1	Não identifiquei nenhum code smell.
14/05/2025 17:55:50		UFMG #26	ConversionException	5	A classe ConversionException aparenta não implementar os métodos de Exception, e só repassa o que foi utilizado nos parâmetros para os seus construtores, para avisar que não foi possível fazer a conversão.
14/05/2025 18:00:28		UFMG #4	TextEditorUtils.java	1	I did not recognize the refused bequest code smell on the code , since there is no explicitly inheritance so that's why I ticked off 1.
14/05/2025 18:01:34		UFMG #43	FrameworkField	4	Apresenta vários @Override da classe que está extendendo, indicando que seu funcionamento não é tão parecido como deveria
14/05/2025 18:01:40		UFMG #9	CellFormulaTagHandler	3	A classe subscreve os métodos da classe herdada, mas acho que isso não classificaria como Refused Bequest, visto que os métodos são utilizados. Acho que talvez teria Long Parameter List nos métodos.
14/05/2025 18:06:29		UFMG #17	HttpRequestMethodsMatcherTest	4	O método implementado está herdando métodos e atributos mas está ignorando eles completamente e implementando seu próprio estilo.
14/05/2025 18:07:03		UFMG #13	IncompleteElementException.java	1	Para mim não está gerando code smells, os métodos estão sendo herdados e usados
14/05/2025 18:07:23		UFMG #2	FrameworkField	2	A classe filho, apesar de herdar da classe mãe, não tira tanto proveito da herança. No caso em que isso ocorre, é um uso mínimo.
14/05/2025 18:08:21		UFMG #30	TransactionMQProducer	5	Subclasse herda DefaultMQProducer mas sobrescreve e ignora partes, adicionando lógica própria.
14/05/2025 18:09:11		UFMG #10	RandomGenerator	3	O código apresenta dois "Override" da classe "mãe", ou seja, da classe que foi herdada. Portantanto, pode haver um badsmell de Refused Bequest. Se houvesse mais informações sobre a classe mãe, poderia afirmar com mais propriedade sobre o bad smell.
14/05/2025 18:09:33		UFMG #27	HttpRequestMethodsMatcherTest	1	Não há evidência de Refused Bequest aqui. A subclasse está aproveitando e especializando a superclasse corretamente, implementando o método abstrato obrigatório, e adicionando testes específicos.
14/05/2025 18:10:30		UFMG #37	AutoLocker	2	Eu não acredito que haja o code smell Refused Bequest na classe AutoLocker, pois esse code smell pressupõe a existência de uma subclasse herdando da superclasse, porém a classe AutoLocker não herda de nenhuma outra. Não tenho certeza se há outro code smell no código, mas acredito que não.
14/05/2025 18:10:59		UFMG #30	FileRefreshableDataSource	5	Subclasse herda de AutoRefreshDataSource mas redefine muitos comportamentos e adiciona nova lógica específica
14/05/2025 18:12:07		UFMG #13	SaTokenException.java	5	Os métodos da classe não utiliza as funções herdadas
14/05/2025 18:12:36		UFMG #24	SMSParsedResult	1	Bem encapsulado ao meu ver
14/05/2025 18:13:52		UFMG #34	ExecutorRouteBusyover	2	Minha confiança é 2. A classe ExecutorRouteBusyover estende ExecutorRouter e sobrescreve o método route. Sem uma visão completa da classe ExecutorRouter e seus métodos (abstratos ou concretos), é difícil afirmar com alta confiança a ocorrência de Refused Bequest. Se ExecutorRouter possuir muitos outros métodos que oferecem funcionalidades de roteamento que ExecutorRouteBusyover ignora ou não utiliza, então o smell estaria presente, mas apenas com o código fornecido, ela parece estar cumprindo o contrato esperado ao implementar route
14/05/2025 18:14:09		UFMG #32	ExecutorRouteBusyover	2	O código apresenta o Long Method, pois o método route realiza várias tarefas (iteração, chamada de API, formatação de resultados) em um único bloco. Não acredito que há Refused Bequest neste caso.
14/05/2025 18:15:11		UFMG #25	ExecutorRouteBusyover	3	Estou dando nota 3 pois: 1)Método grandinho route(...) passa de 40 linhas. Ele faz de tudo: percorre endereços, chama idleBeat, formata a mensagem e decide o retorno. Isso é o Long Method clássico. 2)StringBuffer montando HTML misturado com regra de roteamento. Apresentação + lógica no mesmo lugar atrapalha leitura e testes. 3)Catch genérico engole qualquer exceção e devolve apenas o texto. Fica difícil diferenciar falha de rede, timeout etc. Silenciar erro assim foi citado como mau sinal. 4)Números mágicos: o texto e rótulos de I18n estão espalhados; deveriam ficar em constantes ou no arquivo de internacionalização. Nada crítico, mas vale extrair a parte de formatação e talvez dividir o método.
14/05/2025 18:16:06		UFMG #35	AckSchedulerKey	1	A classe herda métodos de outra porém sempre realiza ações, os métodos herdados não são vazios. Não tem esse code smell.
14/05/2025 18:17:12		UFMG #31	AbortedTransactionException	2	O código não aparenta ter nenhum code smell porque é simples.
14/05/2025 18:17:16		UFMG #19	LexemePath.java	1	The methods seem to be implementing new features and not just only inheriting without a solid reason . That's why I did not see any refused bequest code smell.
14/05/2025 18:18:06		UFMG #40	SMSParsedResult	2	Considerando apenas o contexto da classe, não é possível definir a existência de Refused Bequest especificamente. Caso tivéssemos acesso à classe pai, a classificação seria mais precisa. Porém, considerando as informações que temos, caso a classe pai possua apenas o atributo ParsedResultType, não se trata de um code smell.
14/05/2025 18:19:02		UFMG #8	MethodVisitor	1	O código apresenta uma classe que possui todos os métodos como void e sem nenhum código a ser executado. Logo, para mim a classe é uma data classe, por ser inútil/não ter funcionalidade.
14/05/2025 18:19:12		UFMG #26	InlineSizeHistoVisitable	2	Não consegui compreender o código completamente, mas, aparentemente, os override utilizados alteram os métodos da superclasse corretamente, sem lançamento de exceções para travar os métodos.
14/05/2025 18:20:08		UFMG #31	ApiException	2	A classe herda RuntimeException, e parece utilizar ela corretamente
14/05/2025 18:21:21		UFMG #21	ImmutableEnumMap	2	Não tenho informação da classe pai, então não consigo dizer se há ou não refused bequest.
14/05/2025 18:21:59		UFMG #25	eval-llms-code-smells /ExecutorRouteBusyover	1	Não acredito que tem Refused Bequest nesse código, pois a classe ExecutorRouteBusyover é Filha de ExecutorRoute, porém não conseguimos ver baseado nesse código quais os atributos e métodos da classe ExecutorRoute
14/05/2025 18:23:13		UFMG #12	RedisPriorityScheduler	3	The class RedisPriorityScheduler only call the constructor from its superclass and not another inherited methods. Also, it overrides two methods. It's unknown how many methods and attributes are inherited in total from the superclass RedisScheduler, but if there're many many ones, than there
14/05/2025 18:24:53		UFMG #14	ApiException	3	Não consigo dizer uma vez que não sei as funcionalidades da classe pai.
14/05/2025 18:27:37		UFMG #43	ExcelDataConvertionException	2	Não acredito que tenha a presença de Refused Bequest pois não sobrescreve os métodos e ainda utiliza a superclasse. Por mais que apresente vários novos métodos, não aparenta ser o smell relacionado.
14/05/2025 18:31:54		UFMG #41	DokanyVolume	5	A subclasse sobrescreve quase todos os métodos da classe pai
14/05/2025 18:32:48		UFMG #35	SaTokenException	4	Parece ser o caso desse code smell pois nessa classe que herda de RuntimeException, não usa os métodos herdados(ou usa muito pouco).
14/05/2025 18:33:05		UFMG #28	RelationshipTest	5	A classe RelationshipTest herda de outra classe mas não usa todos os métodos fornecidos por ela, como no método "anyWithIllegalArgumentExceptionThrow". Além disso, altera alguns métodos da classe que extende.
14/05/2025 18:33:49		UFMG #37	btActivatingCollisionAlgorithm	4	Eu acredito que haja o code smell Refused Bequest na classe, pois a metade dos seus métodos é sobrescrita na implementação
14/05/2025 18:34:08		UFMG #5	LexemePath	4	Trata-se de um long class.
14/05/2025 18:35:48		UFMG #36	RelationshipTest	1	Não acredito que há bad smell, porque a classe RelationshipTest herda de AbstractFakerTest e utiliza adequadamente a configuração e os recursos da superclasse., como por exemplo na chamada de super.before() e pelo uso de faker em diversos metodos. Não há indicações de que RelationshipTest esteja ignorando ou "recusando" funcionalidades herdadas de forma a caracterizar o "bad smell" de Refused Bequest.os.
14/05/2025 18:37:16		UFMG #22	RedisPriorityScheduler.java	1	Acredito que não há
14/05/2025 18:37:44		UFMG #12	AckSchedulerKey	3	The class AckSchedulerKey only uses the superclasse's constructor once and overrides two of its methods, which may indicate a Refused Bequest code smell depending on the total amount of attributes and methods in the superclass
14/05/2025 18:40:47		UFMG #37	SMSParsedResult	2	Eu acredito que não há o code smell Refused Bequest na classe, pois ela sobrescreve apenas um método da superclasse
14/05/2025 18:43:06		UFMG #36	InlineSizeHistoVisitable.java	4	Vários cases e funções herdadas de outros métodos
14/05/2025 18:56:07		UFMG #19	MapDeserializer	2	A classe MapDeserializer herda de ContextObjectSerializer, mas não foram encontrados indícios de Refused Bequest. Ela não usa Override em momento nenhum e nem parece repropositar métodos herdados. Pode ser que ela ignore campos herdados entretanto, visto que em seu construtor ela não define nada, apenas gera um novo objeto do tipo da classe, mas isso por si só não parece ser o suficiente para determinar Bad Smell.
27/05/2025 15:24:30		UFMG #15	MapDeserializer	4	A classe MapDeserializer herda de ContextObjectDeserializer, mas recusa seu contrato conceitual, sendo apenas nominal e causa fragilidade no design
04/06/2025 19:41:23		UFOP #8	btActivatingCollisionAlgorithm	4	Todos os métodos da classe, com exceção dos dois construtores e de um método get, fazem uma sobrescrita de métodos da classe pai, alterando seu comportamento.
04/06/2025 19:41:52		UFOP #12	FileRefreshableDataSource	1	Acredito que não tenha um code smell de Refused Bequest, tendo em vista que não encontrei nenhuma função que herdasse alguma função a qual não suportaria. Assim como no repositório de exemplo.

Carimbo de data/hora	Pontuação	Name and Student ID	What is the name of the file you evaluated?	How would you rate your confidence that there is a REFUSED BEQUEST smell in the code?	Mention why (for instance, which code elements) you gave this rating.
04/06/2025 19:46:13		UFOP #27	StubConnection.java	5	da linha 90 até 500 é só override de throws exception
04/06/2025 19:51:55		UFOP #21	HttpRequestMethodsMatcherTest	2	Acredito que tem poucas chances de ser o Refused Bequest, devido ao código somente utilizar get's, ou seja, está mais parecido com o code smell classificado como data class.
04/06/2025 19:53:52		UFOP #23	ImmutableEnumMap	4	Pelos vários Override, e de mudança na classe de "resultado"
04/06/2025 19:57:01		UFOP #13	DefaultDateTypeAdapter.java	5	A classe estende TypeAdapter, mas não a usa. A classe implementa os próprios métodos de conversão de data.
04/06/2025 20:01:46		UFOP #8	DokanyVolume	5	8 dos 10 métodos da classe são sobrescritas de métodos da classe pai, alterando seu comportamento, o que torna a herança desnecessária.
04/06/2025 20:02:15		UFOP #12	Evaluator	5	Sim, há um code smell do tipo Refused Bequest, a classe IndexEvaluator possui algumas subclasses que recusam seus métodos, sendo essas IndexLessThan, IndexGreaterThan, IndexEquals.
04/06/2025 20:05:06		UFOP #24	TransactionMQProducer	4	Na verdade esse código me parece ter Long Parameter List, pois tem um métodos com 5 atributos.
04/06/2025 20:08:00		UFOP #23	PendingTransactionsDialog	5	Ele está herdando mas não o coloca em prática
04/06/2025 20:08:17		UFOP #1	LineTransformationOutputStream	5	A classe LineTransformationOutputStream, altera o comportamento que advém de OutPutStream, realizando um Override em seus métodos, isso caracteriza o Refused Bequest, já que a classe não usa o que é herdado.
04/06/2025 20:09:35		UFOP #25	DefaultDateTypeAdapter	3	ao analisar o código não consegui identificar se realmente a o code smell indicado
04/06/2025 20:10:05		UFOP #10	RedissonTransactionalBucket	5	O código possui um smell do tipo Refused Bequest, uma vez que algumas classes herdam de outras classes mas não utilizam boa parte de seus métodos, pode ser visto na classe RedissonTransactionalBucket<V> que herda de RedissonBucket<V>
04/06/2025 20:11:33		UFOP #11	FileRefreshableDataSource	2	Não parece apresentar refused bequest em sua maioria, entretanto o método readSource() modifica bastante a função o que poderia se caracterizar.
04/06/2025 20:12:38		UFOP #9	RedissonTransactionalBucket	5	Com certeza possui o code smell Refused Bequest, pois há uma subclasse que herda métodos de outra classe, porém os métodos não são utilizados por não serem suportados.
04/06/2025 20:13:10		UFOP #19	LineTransformationOutputStream	4	Não fui capaz de perceber, em nenhum momento, a utilização de métodos provenientes das classes pais pelas classes filhas, portanto acredito estar presente o code smell Refused Bequest.
04/06/2025 20:13:42		UFOP #8	HelpCommand	3	A classe em nanhum momento utiliza de atributos ou comportamentos da classe pai, utilizando mais de classes externas.

04/06/2025 20:14:00		UFOP #22	StubConnection.java	5	Sim, há. Pois há a classe StubConnection herda de outra classe, e sobrescreve um método da classe pai fazendo nenhuma alteração, ou apenas lançando uma excessão.
					Como por exemplo:
					@Override
					public String nativeSQL(String sql) throws SQLException
					{
					return null;
					}

04/06/2025 20:17:40		UFOP #12	HttpHeadRequestTest	1	Acredito que não tenha, pois não achei nenhuma classe herdando de outra.
04/06/2025 20:20:43		UFOP #22	PendingTransactionsDialog.java	1	Não há, pois a subclasse PendingTransactionsDialog usa efetivamente os métodos que está herdando da classe pai
04/06/2025 20:22:15		UFOP #13	RedissonTransactionalBucket.java	5	Embora a classe estenda e implemente muitos métodos de RedissonBucket, ela chama várias delas apenas para lançar uma exceção. É o caso das chamadas moveAsync e migrateAsync, por exemplo.
04/06/2025 20:23:40		UFOP #32	StubConnection	3	A classe StubConnection herda de StubBaseConnection, mas muitos métodos sobrescritos apenas lançam exceções ou retornam valores nulos, sem adicionar lógica útil.
04/06/2025 20:24:09		UFOP #30	HelpCommand	2	A classe HelpCommand possui apenas dois métodos sobrescritos.
04/06/2025 20:24:41		UFOP #21	RedissonTransactionalBucket	2	Acredito que o código tem mais a ver com a classificação de long Parameter List, pois existem metódos que estão exigindo muitos parâmetros que podem comprometer o pleno funcionamento do sistema, bem como acredito que não existem heranças sem coerência no código.
04/06/2025 20:24:42		UFOP #15	DokanyVolume	5	A subclasse sobrescreve um método pai para não fazer nada (ou lançar uma exceção).
04/06/2025 20:27:04		UFOP #18	FrameworkField.java	3	Classe como FrameqorkFiels acessa diretamente atributos de outras classes.
04/06/2025 20:27:42		UFOP #2	DokanyVolume	5	A classe possui uma série de @Override sem uso objetivo.
04/06/2025 20:29:46		UFOP #26	DefaultDateTypeAdapter	1	Não é um caso de Refused Bequest pois a classe usa de fato os métodos herdados, como no verifyDateType onde ele verifica se a condição é valida e retorna o dateType
04/06/2025 20:41:59		UFOP #18	Evaluator.java	1	Código coeso e métodos bem definidos.
04/06/2025 20:43:58		UFOP #29	Evaluator	4	Acredito que tenha refused bequest pois as classes que herdam de Evaluator não utiliza seus métodos.
04/06/2025 20:46:50		UFOP #16	PendingTransactionsDialog	1	O código não apresenta o code smell REFUSED BEQUEST, não aparenta que nenhum método ou atributo foi recusado da classe herdada.
04/06/2025 20:49:09		UFOP #3	HelpCommand	2	método createCommandVO, que é um pouco longo por transformar muitos detalhes de um comando em um formato para exibição. Mas é bem organizado o código e de facil entendimento