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 LONG PARAMETER LIST smell in the code?	Mention why (for instance, which code elements) you gave this rating.
27/04/2025 14:42:50		UFMG #5	UrlConnectionClient.java	1	Não tem nenhum método que realmente faça muito uso de primitivos. O que tem dois é um construtor, então....
28/04/2025 22:19:29		UFMG #2	AttributeNameHistoWalker	3	The constructor has 4 parameters, but I do not think this is a big issue.
29/04/2025 12:40:09		UFMG #30	RequestInformation	1	Único método com parâmetros é o construtor e 5 parâmetros para um método construtor me parece razoável, inclusive pelo fato da classe possuir os 5 atributos referentes à esse parâmetro.
29/04/2025 22:04:38		UFMG #43	PrefixPluginLogger	5	O método logMessage recebe como parâmetros de entrada 5 variáveis de tipos diferentes.
01/05/2025 18:11:05		UFMG #29	AccessTokenModel	2	O construtor AccessTokenModel(String accessToken, String clientId, Object loginId, String scope) tem uma lista de parâmetros aceitável, mas se adicionássemos mais, ficaria demasiadamente longa.
06/05/2025 18:07:35		UFMG #42	RetrofitError	4	Eu achei essa parte do código complexa e confusa. Além de ter muitos paramentros: RetrofitError(String message, String url, Response response, Converter converter, Type successType, boolean networkError, Throwable exception)
06/05/2025 21:05:53		UFMG #39	EntryConfig	3	Trata-se de 6 atributos aparente necessários para classe de configuração. O maior construtor possui atributos para todos eles. Talvez Lombok ajudaria, mas ainda assim os construtores intermediários seriam mantidos.
06/05/2025 21:17:28		UFMG #19	SymbolInfo	4	Estou considerando um dos construtores como sendo code smell. Há formas mais elegantes para implementar esta classe.
06/05/2025 21:25:33		UFMG #19	JobTriggerPoolHelper	4	Métodos trigger e addTrigger com os mesmos parâmetros, um chamando o outro.
06/05/2025 21:43:30		UFMG #19	MethodCallInliner	3	Alguns métodos com 5 parâmetros que aparentemente poderiam ser organizados de forma melhor.
14/05/2025 17:33:01		UFMG #3	EnablePluginCommand.java	1	the maximus n of parameters is 2, not even 3 where theres more room for discussion about the presence of code smell 2 is completely fine to me.
14/05/2025 17:41:45		UFMG #2	AndroidGL20	4	O código possui métodos com a mesma estrutura, o que gera chamadas repetidas para GLE20. Além disso, possui nomes não muito descritivos e a classe acaba violando o principio de responsabilidade única. Por fim, a classe não parece implementar lógicas adicionais e parece mais servir como um pass-through.
14/05/2025 17:42:24		UFMG #6	MethodCallInliner	2	Há um método longo, porém por causa de comentários em sua maioria.
14/05/2025 17:46:12		UFMG #17	PullAPIWrapper	3	As classes estão recebendo muitos parâmetros, o que ocasionou o código a algo de difícil compreensão.
14/05/2025 17:46:39		UFMG #18	ClassLoaderModel	3	Pois ele representa um modelo de domínio anêmico: uma classe que só agrupa dados por meio de getters e setters, sem qualquer comportamento associado, o que tipicamente indica baixa coesão e violação de princípios de orientação a objetos martinfowler. com. Portanto, embora não haja um defeito funcional imediato, esse design sugere a necessidade de extrair comportamentos para dentro do próprio objeto ou repensar o modelo para melhorar encapsulamento e expressividade.
14/05/2025 17:49:21		UFMG #35	AnnotationScanner	4	Eu notei que o método AddListener possui muitos parâmetros, portanto o código possui o Bad Smell: Long Parameter List.
14/05/2025 17:49:24		UFMG #31	ExpandedProductParsedResult.java	5	Os code smells mais claros no código são Large Class e Long Method. A classe possui muitos atributos privados e, consequentemente, o construtor recebe muitos atributos. Por esse motivo, os métodos equals e hashCode também ficam muito extensos.
14/05/2025 17:50:41		UFMG #15	EntryConfig	4	o método `EntryConfig` tem muitas declarações com diferentes parâmetros: `public EntryConfig(String resourceName, int resourceType, EntryType entryType, int acquireCount, Object[] args)`
14/05/2025 17:52:52		UFMG #39	ExpandedProductParsedResult	5	A classe é extremamente grande, possui apenas atributos e seus respectivos gets, e em demasiado. Então o código possui um code smell de Data Class e Large Class.
14/05/2025 17:54:57		UFMG #4	SmsFlashPromotionProductRelationServiceImpl.	4	Long flashPromotionId, Long flashPromotionSessionId . Regarding both parameters as there are 2 methods using those parameters and to initialize the list , it calls another more parameters . So in my opinion , those information could be encapsulated within an object.
14/05/2025 17:55:25		UFMG #13	BeanCopier.java	4	O método de criação de BeanCopier possui muitos parâmetros, poderia ser reduzido em uma classe com opções ou mais métodos
14/05/2025 17:55:54		UFMG #38	MethodVisitor	1	The Long Parameter List code smell is not present in this code. The max number of parameters on the method signatures is 3, which is totally acceptable in my opinion.
14/05/2025 17:56:59		UFMG #3	PrefixPluginLogger.	4	theres several parameters in the method but theres no much other way around the method is small and all of them is part of the message, maybe a data class or a prototype but u would classify this as another code smell so ....
14/05/2025 17:57:21		UFMG #16	DiagnosedStreamCorruptionException	5	Long Parameter List - Não encontrado Refused Bequest - É feito um override do metodo string
14/05/2025 17:57:48		UFMG #6	JobTriggerPoolHelper	4	A classe JobTriggerPoolHelper gerencia diretamente o controle de execução dos jobs, incluindo a escolha entre pools de threads (fastTriggerPool e slowTriggerPool), a execução do trigger e a contagem de timeouts, tudo centralizado em um único método (addTrigger). Isso gera acoplamento excessivo e baixa coesão, dificultando a manutenção e testes. A lógica relacionada ao controle de timeout (como minTim, jobTimeoutCountMap e seus resets) está misturada com a lógica de trigger, o que fere o princípio da responsabilidade única e compromete a clareza do código.
14/05/2025 17:57:55		UFMG #23	PrefixPluginLogger	4	Construtor e logMessage usam listas longas de parâmetros, dificultando manutenção.
14/05/2025 18:00:26		UFMG #36	HttpClientDownloader	2	Esse código tem apenas uma função com uma lista razoavelmente grande de parâmetros, que seriam quarto, mas não acho que isso afeta tanto o rastreio de erros caso ocorra. Então o bad smell é baixo.
14/05/2025 18:02:29		UFMG #37	Finance	3	Dei nota 3 pois: 1) createCountryCodeToBasicBankAccountNumberPatternMap() tem ≈ 200 linhas, ainda que sejam só inserções no Map. Continua sendo um Long Method. 2) creditCard(CreditCardType) (por volta de 40 linhas) executa três responsabilidades 3)calculateIbanChecksum() (por volta de 30 linhas) mistura transformação de caracteres, conversão numérica e cálculo de módulo. Conclusão: A classe, como um todo, é coesa (tudo é "Finance"), e a maioria dos métodos pequenos não apresenta problemas graves; por isso optei por 3 em vez de 4 ou 5.
14/05/2025 18:02:46		UFMG #35	PendingEntry	2	No método public PendingEntry existe um número razoável de parâmetros, mas não acho que se enquadre como um exagero, portanto creio não ser um bad smell.
14/05/2025 18:03:03		UFMG #9	RetrofitError	5	O método RetrofitError certamente tem muitos parâmetros.
14/05/2025 18:03:04		UFMG #18	TreeTypeAdapter	5	Ele recebe nove parâmetros distintos (UUID, AckManager, DisconnectableHub, StoreFactory, HandshakeData, ClientsBox, Transport, CancelableScheduler e Configuration), tornando difícil entender a finalidade de cada um e aumentando o acoplamento entre componentes. Métodos ou construtores com mais de três ou quatro parâmetros costumam indicar que parte da lógica deveria ser extraída para objetos de parâmetro ou agrupada em classes de valor, melhorando a legibilidade e a manutenibilidade do código.
14/05/2025 18:03:48		UFMG #33	RetrofitError	5	O método chamado RetrofitError apresenta 7 parâmetros de entrada, que eu considero muito alto para um método só, caso de Long Parameter List.
14/05/2025 18:04:40		UFMG #7	SmsFlashPromotionProductRelationServiceImpl	4	O código tem muitos parâmetros sendo passados para os métodos
14/05/2025 18:05:19		UFMG #24	AnnotationScanner	1	Codigo limpo e objetivo
14/05/2025 18:05:32		UFMG #18	AttributeNameHistoWalker	3	Ele recebe cinco parâmetros distintos, ultrapassando o limite recomendado de três ou quatro parâmetros e tornando a assinatura difícil de compreender e manter. Contudo, a coerência do método e a natureza infrequente de sua invocação reduzem meu grau de certeza, pois em certos casos essas dependências são inevitáveis e justificadas.
14/05/2025 18:08:21		UFMG #21	SmsFlashPromotionProductRelationServiceImpl	4	Sim, pois o método list apresenta muitos parâmetros
14/05/2025 18:10:32		UFMG #13	MethodCallInliner.java	4	O método CatchBlock, MethodCallInliner e visitMethodInsn possuem muitos parâmetros
14/05/2025 18:11:15		UFMG #20	easyexcel-2.2.11_ExcelDataConvertException	3	A classe possui construtores que recebem quatro parâmetros não triviais, o que pode tornar a criação da exceção um pouco complexa e difícil de manter. Embora esses parâmetros sejam necessários para descrever o erro, uma possível melhoria seria encapsulá-los em um objeto único ou utilizar o padrão Builder para reduzir a quantidade de parâmetros e aumentar a clareza do código.
14/05/2025 18:11:48		UFMG #44	PullAPIWrapper	5	A função pullKernelImpl possui uma grande lista de parametros.
14/05/2025 18:12:10		UFMG #34	Finance	1	Minha confiança é 1. Analisando os métodos da classe Finance, como creditCard(CreditCardType creditCardType) que recebe apenas um parâmetro, ou calculateIbanChecksum(String countryCode, String basicBankAccountNumber) com dois, não se observa o code smell Long Parameter List. Os métodos apresentados possuem listas de parâmetros curtas e concisas, o que é uma boa prática
14/05/2025 18:12:35		UFMG #4	FireBirdMetaModel.java	5	The function genericProcedure , for instance , contains more than 5 parameters which is a good indicator of "long parameter list" code smeel.
14/05/2025 18:13:22		UFMG #31	TreeTypeAdapter	3	O construtor não recebe muitos parâmetros. Mas um code smell que pode estar presente é o Intensive Coupling devido às interações com a Gson
14/05/2025 18:13:49		UFMG #21	ExecutorRouteBusyover	2	Não, pois não existem muitos parâmetros
14/05/2025 18:14:24		UFMG #18	HttpClientDownloader	2	Não há métodos com listas excessivamente grandes de parâmetros. O método com maior número de parâmetros recebe quatro argumentos, que ainda está dentro de um limite razoável quando cada parâmetro tem papel claro e coeso. A maioria dos métodos utiliza no máximo dois parâmetros, bem abaixo do que normalmente dispara esse tipo de cheiro de código. Portanto, há pouca evidência de que seja necessário extrair objetos de parâmetro para melhorar a legibilidade ou a manutenibilidade.
14/05/2025 18:15:01		UFMG #26	TreeTypeAdaptor	2	O construtor da classe pode ser considerado por alguns como tendo uma lista grande de parametros, por haver 5 objetos diferentes. Eu, particularmente, não achei problemático.
14/05/2025 18:15:33		UFMG #32	WebLogAspect	5	Sim, o código apresenta o Long Parameter como um code smell, especialmente no método getParameter. Esse método possui uma lógica complexa para tratar diferentes tipos de parâmetros, o que torna difícil de entender e manter.
14/05/2025 18:16:04		UFMG #10	AndroidGL20	5	Existe o code smell long parameter list, mas também existe o large class, por ser uma classe muito extensa.
14/05/2025 18:16:06		UFMG #17	Lexeme	2	Ele recebe quatro parâmetros primitivos (offset, begin, length e lexemeType). Embora quatro parâmetros já fique próximo ao limite em que vale a pena considerar agrupar em um objeto de valor, cada um deles é simples, bem nomeado e faz parte de um único conceito (definir as características básicas de uma lexema), de modo que não há forte evidência de que a legibilidade ou a manutenibilidade sejam comprometidas a ponto de demandar refatoração imediata.
14/05/2025 18:16:58		UFMG #10	RequestLimiter	1	Nenhuma função possui de fato uma longa lista de parametros, porem, foi possível identificar que nas funções "add" e "increment" existem cadeias de mensagens, onde métodos sao chamados em seguida numa mesma variavel.
14/05/2025 18:17:04		UFMG #33	Lifecycle	5	O código contém apenas um classe com 150 linhas, caso de Large Class. Além disso, o método rewriteHudsonWar tem comentários excessivos (caso de Long Method). O método get tem mais de 50 linhas e apresenta responsabilidades e funções demais, com muitos ELSE e IF, caso de Long Method.
14/05/2025 18:19:56		UFMG #1	ReedSolomonDecoder	2	A classe não possui um numero maior do que 3 parâmetros em nenhum de seus métodos e nem em seu construtor. No entanto, alguns dos métodos definitivamente apresentam complexidade elevada, podendo caracterizar um smell de long method.
14/05/2025 18:20:04		UFMG #28	MethodVisitor	4	A lista de parâmetros de "void visitFieldInsn(int opcode, String owner, String name, String desc);" é longa, e essa lista se repete em "void visitMethodInsn(int opcode, String owner, String name, String desc);". Isso caracteriza um Long Parameter List.
14/05/2025 18:20:12		UFMG #6	AccessTokenModel	4	O código apresenta vários code smells, começando pela quebra de encapsulamento, já que todos os campos são public, expondo detalhes internos e permitindo modificações incontroladas. Há também inicialização inconsistente, pois o construtor não define todos os campos (como expiresTime e refreshExpiresTime), o que pode levar a estados inválidos. O método toString() está incorreto, usando nomes como accessTokenTimeout em vez de expiresTime, gerando uma representação inconsistente. Além disso, há um número mágico (-2) retornado em getExpiresIn() sem explicação clara, prejudicando a legibilidade. O método toLineMap() possui acoplamento direto, acessando campos em vez de usar getters, aumentando a dependência da estrutura interna. Há risco de valores nulos no LinkedHashMap, já que campos como refreshToken podem ser null. O uso de Object para loginId é um tipo genérico desnecessário, podendo causar erros em tempo de execução. O toString() também omite o campo loginId, reduzindo sua utilidade para debug. Por fim, há uma chamada redundante a super(), que é inserida automaticamente pelo Java, tornando sua declaração explícita desnecessária. Esses problemas impactam a manutenibilidade, robustez e clareza do código.
14/05/2025 18:21:54		UFMG #2	CameraInputController	2	Embora a classe CameraInputCobntroller tenha um número significativo de parâmetros de configuração, a maior parte deles está bem agrupada dentro da própria classe e são configuráveis através de variáveis de instância. No entanto, alguns métodos, especialmente os que lidam com a configuração de eventos de toque e rotação, acabam tendo longas listas de parâmetros, como no método process.
14/05/2025 18:22:40		UFMG #27	DiagnosedStreamCorruptionException	1	Não possui Long Parameter List. Construtores simples, parâmetros normais.

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 LONG PARAMETER LIST smell in the code?	Mention why (for instance, which code elements) you gave this rating.
14/05/2025 18:23:16		UFMG #36	DiagnosedStreamCorruptionException	2	O construtor da classe aceita 4 parametros, o que acredito que não seja tantos assim. Apesar disso, acredito que pode aumentar as chances de erro e talvez agrupar os parametros relacionados ao diagnostico, por exemplo, poderia ser melhor. Por isso nao tenho certeza de que realmente não há bad smell
14/05/2025 18:24:20		UFMG #8	EntryConfig	5	Possui 6 métodos cujos os construtores recebem diversos parâmetros de entrada. Além disso, o nome dos métodos é o mesmo caracterizando um shortgun surgery, já que se mudar todos terão que ser modificados
14/05/2025 18:26:47		UFMG #8	AnnotationScanner	5	O método addListener possui 4 parâmetros de entrada
14/05/2025 18:28:22		UFMG #14	FxApplication	5	Os nomes dos métodos e dos objetos são extremamente grandes, dificultando a leitura.
14/05/2025 18:29:36		UFMG #30	FireBirdMetaModel	2	Apesar do método createTableColumnImpl() possuir uma quantidade maior de parâmetros que o usual, isso poderia não ser um problema necessariamente pelo tamanho da tabela justificar que o método seja implementado assim. Contudo, alguns métodos poderiam ser refatorados para reduzir o número de parâmetros, mas não vejo como uma questão urgente.
14/05/2025 18:29:40		UFMG #37	Vault	5	Na classe Vault, há o code smell Long Parameter List, pois o construtor requer oito parâmetros para ser inicializado. Além disso, há o code smell Intensive Coupling, pois a classe é extremamente dependente de outras classes/implementações. Por fim, também observei a presença do code smell Data Class, pois há mais de 150 linhas de métodos que apenas retornam as informações privadas da classe
14/05/2025 18:30:30		UFMG #11	AccessTokenModel	1	O construtor tem um número razoável de parâmetros, nada que atrapalhe o entendimento do funcionamento da classe, tampouco que vá gerar empecilhos na manutenção mais à frente. O restante dos métodos nem parâmetros direito têm.
14/05/2025 18:30:56		UFMG #12	ExpandedProductParsedResult	5	The constructor of the class ExpandedProductParsedResult has more than ten arguments, which configures a Long Parameter List bad smell.
14/05/2025 18:31:29		UFMG #25	eval-llms-code-smells /MockNamingService	1	Na minha opinião o código não possui long parâmetros list, pois o número máximo de parâmetros passados em um método é 5, o que não acho que seja um número muito alto para ser considerado code smell.
14/05/2025 18:32:20		UFMG #5	MethodVisitor	5	Com certeza tem um code smell, esta interface possui um método declarado para cada tipo de classe, o que é loucura e inútil em termos dos benefícios de uma interface. O certo neste caso seria ter um método só mais genérico e então para cada caso em que podemos ter implementações diferentes teríamos uma implementação específica da interface.
14/05/2025 18:33:18		UFMG #41	PendingEntry	3	O construtor da classe PendingEntry possui vários parâmetros. Apesar de isso ser considerado um smell do tipo "Long Parameter List", não vejo tanto problema por se tratar de um construtor.
14/05/2025 18:35:03		UFMG #22	MockNamingService.java	2	Este é o codesmell fornecido
14/05/2025 18:35:05		UFMG #14	eval-llms-code-smells /JobTriggerPoolHelper	4	O código possui long parameter list, pois possui um método com 6 parâmetros, o que considero um valor alto.
14/05/2025 18:35:12		UFMG #12	ExcelDataConvertException	2	The constructors and methods of the ExcelDataConvertException class have five or six parameters at most, which doesn't seems to be long. Thus, probably there's no Long Parameter List bad smell in the code.
14/05/2025 18:35:51		UFMG #26	Configuration	1	Definitivamente não há uma longa lista de parametros para nenhum método.
14/05/2025 18:39:11		UFMG #11	FxApplication	5	Basta ver o construtor de FxApplication para logo enxergar o problema. Ele deve ter mais de 10 parâmetros, o que é inviável na prática de desenvolvimento de software pois torna as chamadas muito verbosas e difíceis de controlar. Claramente um Long Parameter.
14/05/2025 18:45:49		UFMG #40	BatchExecutor	4	Apresenta muitos parâmetros na função doQuery.
27/05/2025 11:43:37		UFMG #32	MailUtil	5	Contém diversos métodos de envio de email com listas longas de parâmetros
04/06/2025 19:42:59		UFOP #17	BinaryRedisPipeline.java	2	não há métodos com parâmetros excessivos
04/06/2025 19:44:37		UFOP #23	FireBirdMetaModel	1	Porque está parecendo que é o Refused Bequest.
04/06/2025 19:47:41		UFOP #25	MapperBuilderAssistant	5	Os elementos do código que indicam uma grande chance de ter o code smell Long parameter list estão nas linhas 244 a 264 e 411 a 425
04/06/2025 19:48:08		UFOP #33	BinaryRedisPipeline.java	4	Na Interface apresentada no arquivo há 450 linhas com diferentes métodos
04/06/2025 19:48:15		UFOP #14	eval-llms-code-smells /RocketMQMessageHandl	5	A classe RocketMQMessageHandler concetra muitas responsabilidades
04/06/2025 19:51:45		UFOP #20	ClassReader.java	2	Acredito que esteja num limite ok de parâmetros nos métodos que constam nessa classe. Nenhum dos métodos passam de 4 parâmetros, talvez essa possa ser uma quantidade mínima que pode não ser usual, mas não deve apresentar grandes riscos para o código.
04/06/2025 19:52:24		UFOP #5	CharacterReader.java	4	Depois da linha 575 temos long parameter em 3 funções. Eg: private static String cacheString(final char[] charBuf, final String[] stringCache, final int start, final int count) { [...] uma característica seria o uso dos mesmos parametros em funções diferentes
04/06/2025 19:53:31		UFOP #11	easyexcel-2.2.11_ExcelDataConvertException	1	Não se caracteriza um long parameter list já que os parâmetros estão na quantidade necessária para a classe. Entretanto conta com características de um data class.
04/06/2025 19:55:19		UFOP #24	PullAPIWrapper	1	Porque os métodos e construtores presentes nesse código não possuem muitos parâmetros
04/06/2025 19:57:11		UFOP #32	RouterNanoHTTPD	3	Existe métodos com múltiplos parâmetros, a maior parte das interações utiliza objetos como UriResource e IHTTPSession que encapsulam esses dados.
04/06/2025 19:59:38		UFOP #11	ProxyDatabaseMetaData	1	O código não conta com muitos parâmetros. Contém grande quatidade de métodos se enquadrando em um large class.
04/06/2025 20:00:18		UFOP #32	RuleContainer	1	O único método que chega perto é apply, que recebe 4 parâmetros, todos necessários e bem justificados. Não há indícios fortes de má prática.
04/06/2025 20:02:24		UFOP #26	MapperBuilderAssistant	5	Acredito que seja um code smell por causa da quantidade de parâmetros que são passados nos métodos: buildResultMapping, addMappedStatement,addMappedStatement
04/06/2025 20:05:07		UFOP #12	MapperBuilderAssistant	5	Definitivamente possui um code smell do tipo Long Parameter List, pois há diversas funções que recebem mais de 6 ou 7 parametos. Alguns exemplos ParameterMapping e useNewCache
04/06/2025 20:11:35		UFOP #15	RuleContainer	3	Somente em um método identificado uso de parâmetros em excesso.
04/06/2025 20:14:39		UFOP #15	ExcelDataConvertException	5	Métodos da classe recebendo uma enorme lista de paramêtros. Considerado um forte indicador de code smell.
04/06/2025 20:18:35		UFOP #27	ClassWriter	4	Possui alguns métodos com uma quantidade excessiva de parâmetros, que acabam tornando o método um pouco mais dependente.
04/06/2025 20:18:39		UFOP #23	Preconditions	1	Não recebe um número excessivo de parâmetro
04/06/2025 20:20:32		UFOP #6	MapperBuilderAssistant	5	Inúmeros métodos contém um número exagerado de parâmetros instanciados, deixando seu entendimento e manutenção extremamente complicados.
04/06/2025 20:20:38		UFOP #8	Preconditions	4	Os métodos no geral contêm de 4 a 6 argumentos, o que leva a indicar uma longa lista de parâmetros.
04/06/2025 20:20:43		UFOP #14	eval-llms-code-smells /CachingExecutor.java	5	O método ensureNoOutParams() parece ter mais a ver com validação de MappedStatement do que com caching, ele que poderia estar em outra classe.
04/06/2025 20:23:59		UFOP #14	eval-llms-code-smells /BinaryRedisPipeline.java	4	a interface tem muitos métodos
04/06/2025 20:25:17		UFOP #26	RouterNanoHTTPD	5	É um caso de Long Parameter List por causa da Interface UriResponder e UriResource que contem diversos parâmetros. Os metodos precisam de 4 ou mais parâmetros, o que acaba dificultando a leitura, testes e manutenção.
04/06/2025 20:25:46		UFOP #16	AllMembersSupplier	2	A classe não possui muitos parametros, ela é uma herença de AllMembersSupplier, não sei os atributos dessa classe, mas acredito que não sejam muitos tambem.
04/06/2025 20:26:39		UFOP #14	eval-llms-code-smells /BinaryRedisPipeline.java	3	A classe implementa o padrão envolve um Executor delegado
04/06/2025 20:29:36		UFOP #20	/blob/main/Lorem	1	Mesmo que alguns métodos tenham uma quantidade maior de paremetros, todos são utilizados dentro do mesmo, logo ele se torna fundamental. Caso alguns dos parametros não fossem utilizados , seria o caso de parametros além.
04/06/2025 20:29:40		UFOP #4	Preconditions	1	Atribui a nota 1, pois o que eu pude identificar foram diversos comentários, dando a impressão de que existem mais comentários do que linhas de código. Logo, me parece ser um code smell do tipo Long Method.
04/06/2025 20:30:07		UFOP #13	RuleContainer.java	1	O método com mais parâmetros da classe possui apenas quatro.
04/06/2025 20:34:11		UFOP #18	CharacterReader.java	3	Metodos dependem de muitos dados de outras classes, possivel que haja code smells.
04/06/2025 20:34:37		UFOP #25	Lorem	2	o código não possui uma lista extensiva de paramentos, mas possui muitos métodos
04/06/2025 20:36:48		UFOP #2	BinaryRedisPipeline	5	A classe possui uma quantidade grande de parâmetros, sendo difícil de compreende-la.