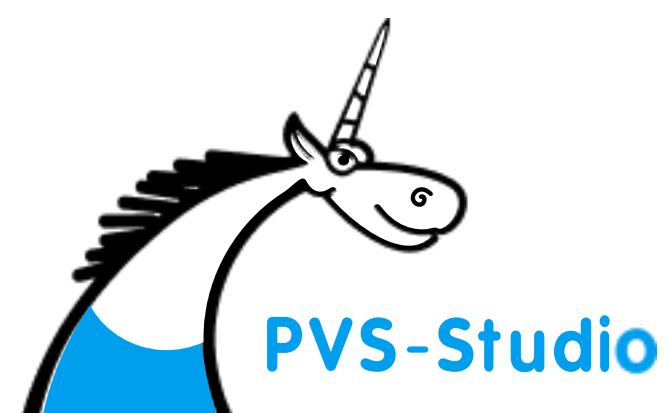


# ЧИСТЫЙ КОД

Когда красота работает на  
надёжность



**Валерий Филатов**  
Developer Advocate



# ВАЛЕРИЙ ФИЛАТОВ

DEVELOPER ADVOCATE

- Занимаюсь поддержкой сборочной инфраструктуры и разработкой инструментов для разработчиков.
- Рассказываю про технологии статического анализа и не только в статьях и на различных мероприятиях.

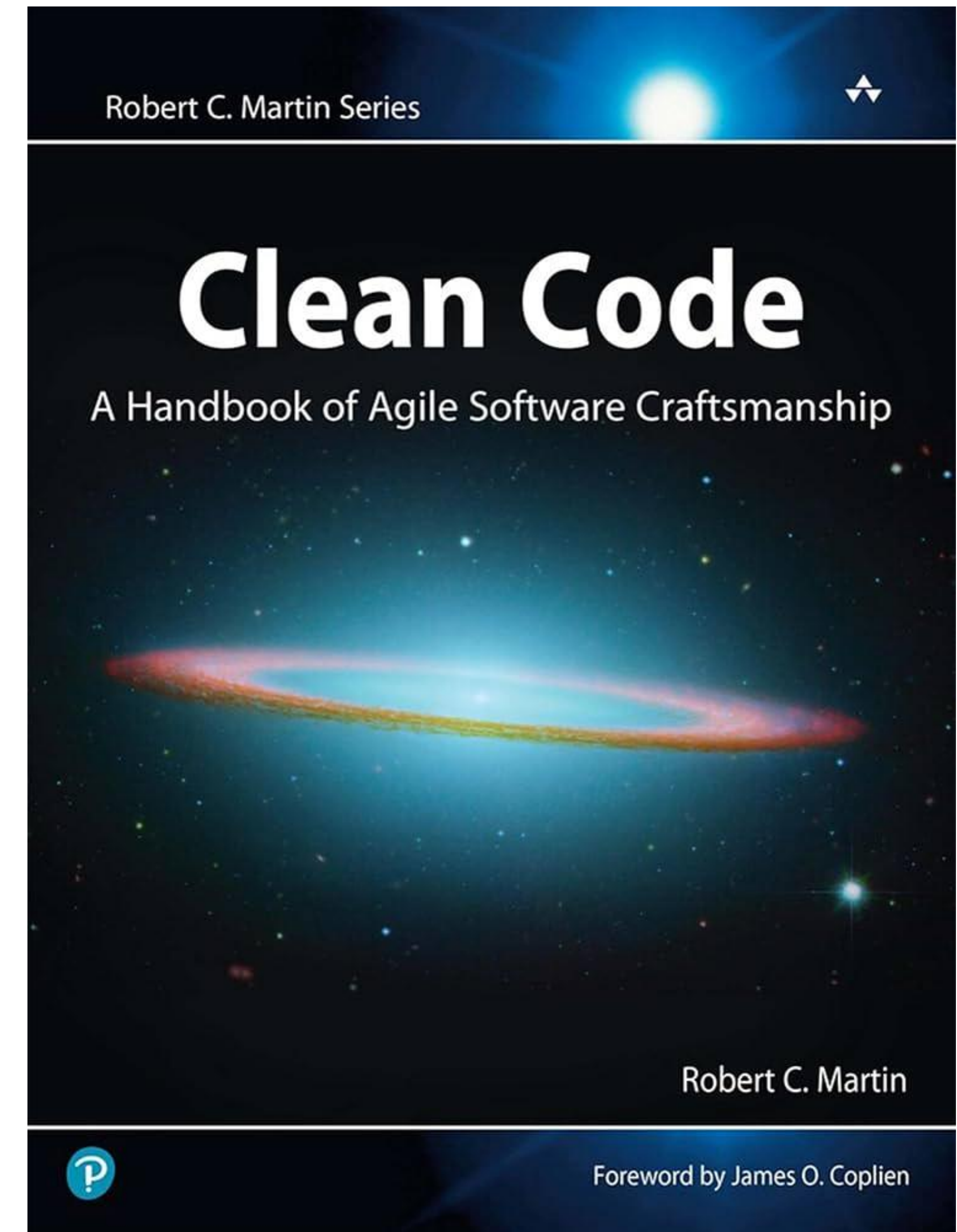
Люблю Python и чистый код (да, это иногда сочетается).



# «ЧИСТЫЙ КОД»

- Практическое руководство по написанию чистого, понятного и поддерживаемого кода.
- Акцент на важности качества кода для долгосрочной разработки ПО.
- Книга адресована разработчикам всех уровней, от новичков до опытных.

Есть [GitHub репозиторий](#) с выжимкой основных пунктов.



«Я люблю, чтобы мой код был элегантным и эффективным. Логика должна быть достаточно прямолинейной, чтобы ошибкам было трудно спрятаться; зависимости – минимальными, чтобы упростить сопровождение; обработка ошибок – полной в соответствии с выработанной стратегией; а производительность – близкой к оптимальной, чтобы не искушать людей загрязнять код беспринципными оптимизациями. Чистый код хорошо решает одну задачу.»

## БЬЁРН СТРАУСТРУП

Создатель C++,  
автор книги «The C++ Programming Language»





# «CODE SMELLS»

Признаки того, что в вашем  
коде что-то не так



# CODE SMELLS

- **Rigidity** - программное обеспечение сложно изменить. Небольшое изменение вызывает цепь последующих изменений.



# CODE SMELLS

- **Rigidity** - программное обеспечение сложно изменить. Небольшое изменение вызывает цепь последующих изменений.
- **Fragility** - программное обеспечение выходит из строя во многих местах из-за одного единственного изменения.



# CODE SMELLS

- **Rigidity** - программное обеспечение сложно изменить. Небольшое изменение вызывает цепь последующих изменений.
- **Fragility** - программное обеспечение выходит из строя во многих местах из-за одного единственного изменения.
- **Immobility** - вы не можете повторно использовать части кода в других проектах из-за связанных с этим рисков и больших затрат.





# CODE SMELLS

- **Rigidity** - программное обеспечение сложно изменить. Небольшое изменение вызывает цепь последующих изменений.
- **Fragility** - программное обеспечение выходит из строя во многих местах из-за одного единственного изменения.
- **Immobility** - вы не можете повторно использовать части кода в других проектах из-за связанных с этим рисков и больших затрат.
- **Opacity** - код сложно понять.



# CODE SMELLS

- **Rigidity** - программное обеспечение сложно изменить. Небольшое изменение вызывает цепь последующих изменений.
- **Fragility** - программное обеспечение выходит из строя во многих местах из-за одного единственного изменения.
- **Immobility** - вы не можете повторно использовать части кода в других проектах из-за связанных с этим рисков и больших затрат.
- **Opacity** - код сложно понять.
- **Ненужная сложность.**



# CODE SMELLS

- **Rigidity** - программное обеспечение сложно изменить. Небольшое изменение вызывает цепь последующих изменений.
- **Fragility** - программное обеспечение выходит из строя во многих местах из-за одного единственного изменения.
- **Immobility** - вы не можете повторно использовать части кода в других проектах из-за связанных с этим рисков и больших затрат.
- **Opacity** - код сложно понять.
- **Ненужная сложность.**
- **Ненужное повторение.**



# БОЛЬШИЕ ГОРОДА

Функции-переростки





```
public ClusterSettings(Config config, string systemName)
{
    //TODO: Requiring!
    var clusterConfig = config.GetConfig("akka.cluster");
    if (clusterConfig?.GetConfig("failure-detector") == null)
        throw new ConfigurationException($"Failed to instantiate {nameof(ClusterSettings)}: Configuration does not contain `akka.cluster` node. Did you forgot to set the 'akka.cluster.provider' HOCON property to 'cluster'?");

    LogInfoVerbose = clusterConfig.GetBoolean("log-info-verbose", false);
    LogInfo = LogInfoVerbose || clusterConfig.GetBoolean("log-info", false);
    _failureDetectorConfig = clusterConfig.GetConfig("failure-detector");
    FailureDetectorImplementationClass = _failureDetectorConfig.GetString("implementation-class", null);
    HeartbeatInterval = _failureDetectorConfig.GetTimeSpan("heartbeat-interval", null);
    HeartbeatExpectedResponseAfter = _failureDetectorConfig.GetTimeSpan("expected-response-after", null);
    MonitoredByNrOfMembers = _failureDetectorConfig.GetInt("monitored-by-nr-of-members", 0);

    SeedNodes = clusterConfig.GetStringList("seed-nodes", new string[] { }).Select(Address.Parse).ToImmutableList();
    SeedNodeTimeout = clusterConfig.GetTimeSpan("seed-node-timeout", null);
    RetryUnsuccessfulJoinAfter = clusterConfig.GetTimeSpanWithOffSwitch("retry-unsuccessful-join-after");
    ShutdownAfterUnsuccessfulJoinSeedNodes = clusterConfig.GetTimeSpanWithOffSwitch("shutdown-after-unsuccessful-join-seed-nodes");
    PeriodicTasksInitialDelay = clusterConfig.GetTimeSpan("periodic-tasks-initial-delay", null);
    GossipInterval = clusterConfig.GetTimeSpan("gossip-interval", null);
    GossipTimeToLive = clusterConfig.GetTimeSpan("gossip-time-to-live", null);
    LeaderActionsInterval = clusterConfig.GetTimeSpan("leader-actions-interval", null);
    UnreachableNodesReaperInterval = clusterConfig.GetTimeSpan("unreachable-nodes-reaper-interval", null);
    PublishStatsInterval = clusterConfig.GetTimeSpanWithOffSwitch("publish-stats-interval");

    var key = "down-removal-margin";
    var useDownRemoval = clusterConfig.GetString(key, "");
    DownRemovalMargin =
        (
            useDownRemoval.ToLowerInvariant().Equals("off") ||
            useDownRemoval.ToLowerInvariant().Equals("false") ||
            useDownRemoval.ToLowerInvariant().Equals("no")
        ) ? TimeSpan.Zero :
        clusterConfig.GetTimeSpan("down-removal-margin", null);

#pragma warning disable CS0618
    AutoDownUnreachableAfter = clusterConfig.GetTimeSpanWithOffSwitch("auto-down-unreachable-after");
#pragma warning restore CS0618

    Roles = clusterConfig.GetStringList("roles", new string[] { }).ToImmutableHashSet();
    AppVersion = Util.AppVersion.Create(clusterConfig.GetString("app-version"));

    MinNrOfMembers = clusterConfig.GetInt("min-nr-of-members", 0);

    _useDispatcher = clusterConfig.GetString("use-dispatcher", null);
    if (string.IsNullOrEmpty(_useDispatcher)) _useDispatcher = Dispatchers.InternalDispatcherId;
    GossipDifferentViewProbability = clusterConfig.GetDouble("gossip-different-view-probability", 0);
    ReduceGossipDifferentViewProbability = clusterConfig.GetInt("reduce-gossip-different-view-probability", 0);
    SchedulerTickDuration = clusterConfig.GetTimeSpan("scheduler.tick-duration", null);
    SchedulerTicksPerWheel = clusterConfig.GetInt("scheduler.ticks-per-wheel", 0);

    MinNrOfMembersOfRole = clusterConfig.GetConfig("role").Root.GetObject().Items
        .ToImmutableDictionary(kv => kv.Key, kv => kv.Value.GetObject().GetKey("min-nr-of-members").GetInt());

    VerboseHeartbeatLogging = clusterConfig.GetBoolean("debug.verbose-heartbeat-logging", false);
    VerboseGossipReceivedLogging = clusterConfig.GetBoolean("debug.verbose-receive-gossip-logging", false);

    var downingProviderClassName = clusterConfig.GetString("downing-provider-class", null);
    if (!string.IsNullOrEmpty(downingProviderClassName))
        DowningProviderType = Type.GetType(downingProviderClassName, true);
#pragma warning disable CS0618
    else if (AutoDownUnreachableAfter.HasValue)
#pragma warning restore CS0618
        DowningProviderType = typeof(AutoDowning);
    else
        DowningProviderType = typeof(NoDowning);

    RunCoordinatedShutdownWhenDown = clusterConfig.GetBoolean("run-coordinated-shutdown-when-down", false);

    TimeSpan GetWeaklyUpDuration()
    {
        var cKey = "allow-weakly-up-members";
        switch (clusterConfig.GetString(cKey, string.Empty)
            .ToLowerInvariant())
        {
            case "off":
                return TimeSpan.Zero;
            case "on":
                return TimeSpan.FromSeconds(7); // for backwards compatibility when it wasn't a duration
            default:
                var val = clusterConfig.GetTimeSpan(cKey, TimeSpan.FromSeconds(7));
                if (!(val > TimeSpan.Zero))
                    throw new ConfigurationException($"Valid settings for [akka.cluster.{cKey}] are 'off', 'on', or a timespan greater than 0s. Received [{val}]");
                return val;
        }
    }

    WeaklyUpAfter = GetWeaklyUpDuration();

    UseLegacyHeartbeatMessage = clusterConfig.GetBoolean("use-legacy-heartbeat-message", false);
}
```



```
public ClusterSettings(Config config, string systemName)
{
    //TODO: Requiring!
    var clusterConfig = config.GetConfig("akka.cluster");
    if (clusterConfig?.GetConfig("failure-detector") == null)
        throw new ConfigurationException($"Failed to instantiate {nameof(ClusterSettings)}: Configuration does not contain `akka.cluster` node. Did you forgot to set the 'akka.cluster.provider' HOCON property to 'cluster'?");

    LogInfoVerbose = clusterConfig.GetBoolean("log-info-verbose", false);
    LogInfo = LogInfoVerbose || clusterConfig.GetBoolean("log-info", false);
    _failureDetectorConfig = clusterConfig.GetConfig("failure-detector");
    FailureDetectorImplementationClass = _failureDetectorConfig.GetString("implementation-class", null);
    HeartbeatInterval = _failureDetectorConfig.GetTimeSpan("heartbeat-interval", null);
    HeartbeatExpectedResponseAfter = _failureDetectorConfig.GetTimeSpan("expected-response-after", null);
    MonitoredByNrOfMembers = _failureDetectorConfig.GetInt("monitored-by-nr-of-members", 0);

    SeedNodes = clusterConfig.GetStringList("seed-nodes", new string[] { }).Select(Address.Parse).ToImmutableList();
    SeedNodeTimeout = clusterConfig.GetTimeSpan("seed-node-timeout", null);
    RetryUnsuccessfulJoinAfter = clusterConfig.GetTimeSpanWithOffSwitch("retry-unsuccessful-join-after");
    ShutdownAfterUnsuccessfulJoinSeedNodes = clusterConfig.GetTimeSpanWithOffSwitch("shutdown-after-unsuccessful-join-seed-nodes");
    PeriodicTasksInitialDelay = clusterConfig.GetTimeSpan("periodic-tasks-initial-delay", null);
    GossipInterval = clusterConfig.GetTimeSpan("gossip-interval", null);
    GossipTimeToLive = clusterConfig.GetTimeSpan("gossip-time-to-live", null);
    LeaderActionsInterval = clusterConfig.GetTimeSpan("leader-actions-interval", null);
    UnreachableNodesReaperInterval = clusterConfig.GetTimeSpan("unreachable-nodes-reaper-interval", null);
    PublishStatsInterval = clusterConfig.GetTimeSpanWithOffSwitch("publish-stats-interval");

    var key = "down-removal-margin";
    var useDownRemoval = clusterConfig.GetString(key, "");
    DownRemovalMargin =
        (
            useDownRemoval.ToLowerInvariant().Equals("off") ||
            useDownRemoval.ToLowerInvariant().Equals("false") ||
            useDownRemoval.ToLowerInvariant().Equals("no")
        ) ? TimeSpan.Zero :
        clusterConfig.GetTimeSpan("down-removal-margin", null);

#pragma warning disable CS0618
    AutoDownUnreachableAfter = clusterConfig.GetTimeSpanWithOffSwitch("auto-down-unreachable-after");
#pragma warning restore CS0618

    Roles = clusterConfig.GetStringList("roles", new string[] { }).ToImmutableHashSet();
    AppVersion = Util.AppVersion.Create(clusterConfig.GetString("app-version"));

    MinNrOfMembers = clusterConfig.GetInt("min-nr-of-members", 0);

    _useDispatcher = clusterConfig.GetString("use-dispatcher", null);
    if (string.IsNullOrEmpty(_useDispatcher)) _useDispatcher = Dispatchers.InternalDispatcherId;
    GossipDifferentViewProbability = clusterConfig.GetDouble("gossip-different-view-probability", 0);
    ReduceGossipDifferentViewProbability = clusterConfig.GetInt("reduce-gossip-different-view-probability", 0);
    SchedulerTickDuration = clusterConfig.GetTimeSpan("scheduler.tick-duration", null);
    SchedulerTicksPerWheel = clusterConfig.GetInt("scheduler.ticks-per-wheel", 0);

    MinNrOfMembersOfRole = clusterConfig.GetConfig("role").Root.GetObject().Items
        .ToImmutableDictionary(kv => kv.Key, kv => kv.Value.GetObject().GetKey("min-nr-of-members").GetInt());

    VerboseHeartbeatLogging = clusterConfig.GetBoolean("debug.verbose-heartbeat-logging", false);
    VerboseGossipReceivedLogging = clusterConfig.GetBoolean("debug.verbose-receive-gossip-logging", false);

    var downingProviderClassName = clusterConfig.GetString("downing-provider-class", null);
    if (!string.IsNullOrEmpty(downingProviderClassName))
        DowningProviderType = Type.GetType(downingProviderClassName, true);
#pragma warning disable CS0618
    else if (AutoDownUnreachableAfter.HasValue)
#pragma warning restore CS0618
        DowningProviderType = typeof(AutoDowning);
    else
        DowningProviderType = typeof(NoDowning);

    RunCoordinatedShutdownWhenDown = clusterConfig.GetBoolean("run-coordinated-shutdown-when-down", false);

    TimeSpan GetWeaklyUpDuration()
    {
        var cKey = "allow-weakly-up-members";
        switch (clusterConfig.GetString(cKey, string.Empty)
            .ToLowerInvariant())
        {
            case "off":
                return TimeSpan.Zero;
            case "on":
                return TimeSpan.FromSeconds(7); // for backwards compatibility when it wasn't a duration
            default:
                var val = clusterConfig.GetTimeSpan(cKey, TimeSpan.FromSeconds(7));
                if (!val > TimeSpan.Zero)
                    throw new ConfigurationException($"Valid settings for [akka.cluster.{cKey}] are 'off', 'on', or a timespan greater than 0s. Received [{val}]");
                return val;
        }
    }

    WeaklyUpAfter = GetWeaklyUpDuration();

    UseLegacyHeartbeatMessage = clusterConfig.GetBoolean("use-legacy-heartbeat-message", false);
}
```





```
public ClusterSettings(Config config, string systemName)
{
    //TODO: Requiring!
    var clusterConfig = config.GetConfig("akka.cluster");
    if (clusterConfig?.GetConfig("failure-detector") == null)
        throw new ConfigurationException($"Failed to instantiate {nameof(ClusterSettings)}: Configuration does not contain `akka.cluster` node. Did you forgot to set the 'akka.cluster.provider' HOCON property to 'cluster'?");

    LogInfoVerbose = clusterConfig.GetBoolean("log-info-verbose", false);
    LogInfo = LogInfoVerbose || clusterConfig.GetBoolean("log-info", false);
    _failureDetectorConfig = clusterConfig.GetConfig("failure-detector");
    FailureDetectorImplementationClass = _failureDetectorConfig.GetString("implementation-class", null);
    HeartbeatInterval = _failureDetectorConfig.GetTimeSpan("heartbeat-interval", null);
    HeartbeatExpectedResponseAfter = _failureDetectorConfig.GetTimeSpan("expected-response-after", null);
    MonitoredByNrOfMembers = _failureDetectorConfig.GetInt("monitored-by-nr-of-members", 0);

    SeedNodes = clusterConfig.GetStringList("seed-nodes", new string[] { }).Select(Address.Parse).ToImmutableList();
    SeedNodeTimeout = clusterConfig.GetTimeSpan("seed-node-timeout", null);
    RetryUnsuccessfulJoinAfter = clusterConfig.GetTimeSpanWithOffSwitch("retry-unsuccessful-join-after");
    ShutdownAfterUnsuccessfulJoinSeedNodes = clusterConfig.GetTimeSpanWithOffSwitch("shutdown-after-unsuccessful-join-seed-nodes");
    PeriodicTasksInitialDelay = clusterConfig.GetTimeSpan("periodic-tasks-initial-delay", null);
    GossipInterval = clusterConfig.GetTimeSpan("gossip-interval", null);
    GossipTimeToLive = clusterConfig.GetTimeSpan("gossip-time-to-live", null);
    LeaderActionsInterval = clusterConfig.GetTimeSpan("leader-actions-interval", null);
    UnreachableNodesReaperInterval = clusterConfig.GetTimeSpan("unreachable-nodes-reaper-interval", null);
    PublishStatsInterval = clusterConfig.GetTimeSpanWithOffSwitch("publish-stats-interval");

    var key = "down-removal-margin";
    var useDownRemoval = clusterConfig.GetString(key, "");
    DownRemovalMargin =
        (
            useDownRemoval.ToLowerInvariant().Equals("off") ||
            useDownRemoval.ToLowerInvariant().Equals("false") ||
            useDownRemoval.ToLowerInvariant().Equals("no")
        ) ? TimeSpan.Zero :
        clusterConfig.GetTimeSpan("down-removal-margin", null);

#pragma warning disable CS0618
    AutoDownUnreachableAfter = clusterConfig.GetTimeSpanWithOffSwitch("auto-down-unreachable-after");
#pragma warning restore CS0618

    Roles = clusterConfig.GetStringList("roles", new string[] { }).ToImmutableHashSet();
    AppVersion = Util.AppVersion.Create(clusterConfig.GetString("app-version"));

    MinNrOfMembers = clusterConfig.GetInt("min-nr-of-members", 0);

    _useDispatcher = clusterConfig.GetString("use-dispatcher", null);
    if (string.IsNullOrEmpty(_useDispatcher)) _useDispatcher = Dispatchers.InternalDispatcherId;
    GossipDifferentViewProbability = clusterConfig.GetDouble("gossip-different-view-probability", 0);
    ReduceGossipDifferentViewProbability = clusterConfig.GetInt("reduce-gossip-different-view-probability", 0);
    SchedulerTickDuration = clusterConfig.GetTimeSpan("scheduler.tick-duration", null);
    SchedulerTicksPerWheel = clusterConfig.GetInt("scheduler.ticks-per-wheel", 0);

    MinNrOfMembersOfRole = clusterConfig.GetConfig("role").Root.GetObject().Items
        .ToImmutableDictionary(kv => kv.Key, kv => kv.Value.GetObject().GetKey("min-nr-of-members").GetInt());

    VerboseHeartbeatLogging = clusterConfig.GetBoolean("debug.verbose-heartbeat-logging", false);
    VerboseGossipReceivedLogging = clusterConfig.GetBoolean("debug.verbose-receive-gossip-logging", false);

    var downingProviderClassName = clusterConfig.GetString("downing-provider-class", null);
    if (!string.IsNullOrEmpty(downingProviderClassName))
        DowningProviderType = Type.GetType(downingProviderClassName, true);
#pragma warning disable CS0618
    else if (AutoDownUnreachableAfter.HasValue)
#pragma warning restore CS0618
        DowningProviderType = typeof(AutoDowning);
    else
        DowningProviderType = typeof(NoDowning);

    RunCoordinatedShutdownWhenDown = clusterConfig.GetBoolean("run-coordinated-shutdown-when-down", false);

    TimeSpan GetWeaklyUpDuration()
    {
        var cKey = "allow-weakly-up-members";
        switch (clusterConfig.GetString(cKey, string.Empty)
            .ToLowerInvariant())
        {
            case "off":
                return TimeSpan.Zero;
            case "on":
                return TimeSpan.FromSeconds(7); // for backwards compatibility when it wasn't a duration
            default:
                var val = clusterConfig.GetTimeSpan(cKey, TimeSpan.FromSeconds(7));
                if (!val > TimeSpan.Zero)
                    throw new ConfigurationException($"Valid settings for [akka.cluster.{cKey}] are 'off', 'on', or a timespan greater than 0s. Received [{val}]");
                return val;
        }
    }

    WeaklyUpAfter = GetWeaklyUpDuration();

    UseLegacyHeartbeatMessage = clusterConfig.GetBoolean("use-legacy-heartbeat-message", false);
}
```

```
MinNrOfMembersOfRole =
clusterConfig
    .GetConfig("role")
    .Root.GetObject()
    .Items
    .ToImmutableDictionary(
        kv => kv.Key,
        kv => kv.Value.GetObject().GetKey(....).GetInt()
    );
```

```
MinNrOfMembersOfRole =  
    clusterConfig  
        .GetConfig("role")  
        .Root.GetObject()  
        .Items  
        .ToImmutableDictionary(  
            kv => kv.Key,  
            kv => kv.Value.GetObject().GetKey(...).GetInt()  
        );
```



```
MinNrOfMembersOfRole =  
    clusterConfig  
        .GetConfig("role")  
        .Root.GetObject()  
        .Items  
        .ToImmutableDictionary(  
            kv => kv.Key,  
            kv => kv.Value.GetObject().GetKey(...).GetInt()  
        );
```

**V3080** Possible null dereference of method return value. Consider inspecting: GetObject().

```
MinNrOfMembersOfRole =  
    clusterConfig  
        .GetConfig("role")  
        .Root.GetObject()  
        .Items  
        .ToImmutableDictionary(  
            kv => kv.Key,  
            kv => kv.Value.GetObject()?.GetKey(...).GetInt()  
        );
```

~~V3080 Possible null dereference of method return value. Consider inspecting: GetObject().~~





```
public ClusterSettings(Config config, string systemName)
{
    //TODO: Requiring!
    var clusterConfig = config.GetConfig("akka.cluster");
    if (clusterConfig?.GetConfig("failure-detector") == null)
        throw new ConfigurationException($"Failed to instantiate {nameof(ClusterSettings)}: Configuration does not contain `akka.cluster` node. Did you forgot to set the 'akka.cluster.provider' HOCON property to 'cluster'?");

    LogInfoVerbose = clusterConfig.GetBoolean("log-info-verbose", false);
    LogInfo = LogInfoVerbose || clusterConfig.GetBoolean("log-info", false);
    _failureDetectorConfig = clusterConfig.GetConfig("failure-detector");
    FailureDetectorImplementationClass = _failureDetectorConfig.GetString("implementation-class", null);
    HeartbeatInterval = _failureDetectorConfig.GetTimeSpan("heartbeat-interval", null);
    HeartbeatExpectedResponseAfter = _failureDetectorConfig.GetTimeSpan("expected-response-after", null);
    MonitoredByNrOfMembers = _failureDetectorConfig.GetInt("monitored-by-nr-of-members", 0);

    SeedNodes = clusterConfig.GetStringList("seed-nodes", new string[] { }).Select(Address.Parse).ToImmutableList();
    SeedNodeTimeout = clusterConfig.GetTimeSpan("seed-node-timeout", null);
    RetryUnsuccessfulJoinAfter = clusterConfig.GetTimeSpanWithOffSwitch("retry-unsuccessful-join-after");
    ShutdownAfterUnsuccessfulJoinSeedNodes = clusterConfig.GetTimeSpanWithOffSwitch("shutdown-after-unsuccessful-join-seed-nodes");
    PeriodicTasksInitialDelay = clusterConfig.GetTimeSpan("periodic-tasks-initial-delay", null);
    GossipInterval = clusterConfig.GetTimeSpan("gossip-interval", null);
    GossipTimeToLive = clusterConfig.GetTimeSpan("gossip-time-to-live", null);
    LeaderActionsInterval = clusterConfig.GetTimeSpan("leader-actions-interval", null);
    UnreachableNodesReaperInterval = clusterConfig.GetTimeSpan("unreachable-nodes-reaper-interval", null);
    PublishStatsInterval = clusterConfig.GetTimeSpanWithOffSwitch("publish-stats-interval");

    var key = "down-removal-margin";
    var useDownRemoval = clusterConfig.GetString(key, "");
    DownRemovalMargin =
        (
            useDownRemoval.ToLowerInvariant().Equals("off") ||
            useDownRemoval.ToLowerInvariant().Equals("false") ||
            useDownRemoval.ToLowerInvariant().Equals("no")
        ) ? TimeSpan.Zero :
        clusterConfig.GetTimeSpan("down-removal-margin", null);

#pragma warning disable CS0618
    AutoDownUnreachableAfter = clusterConfig.GetTimeSpanWithOffSwitch("auto-down-unreachable-after");
#pragma warning restore CS0618

    Roles = clusterConfig.GetStringList("roles", new string[] { }).ToImmutableHashSet();
    AppVersion = Util.AppVersion.Create(clusterConfig.GetString("app-version"));

    MinNrOfMembers = clusterConfig.GetInt("min-nr-of-members", 0);

    _useDispatcher = clusterConfig.GetString("use-dispatcher", null);
    if (string.IsNullOrEmpty(_useDispatcher)) _useDispatcher = Dispatchers.InternalDispatcherId;
    GossipDifferentViewProbability = clusterConfig.GetDouble("gossip-different-view-probability", 0);
    ReduceGossipDifferentViewProbability = clusterConfig.GetInt("reduce-gossip-different-view-probability", 0);
    SchedulerTickDuration = clusterConfig.GetTimeSpan("scheduler.tick-duration", null);
    SchedulerTicksPerWheel = clusterConfig.GetInt("scheduler.ticks-per-wheel", 0);

    MinNrOfMembersOfRole = clusterConfig.GetConfig("role").Root.GetObject().Items
        .ToImmutableDictionary(kv => kv.Key, kv => kv.Value.GetObject().GetKey("min-nr-of-members").GetInt());

    VerboseHeartbeatLogging = clusterConfig.GetBoolean("debug.verbose-heartbeat-logging", false);
    VerboseGossipReceivedLogging = clusterConfig.GetBoolean("debug.verbose-receive-gossip-logging", false);

    var downingProviderClassName = clusterConfig.GetString("downing-provider-class", null);
    if (!string.IsNullOrEmpty(downingProviderClassName))
        DowningProviderType = Type.GetType(downingProviderClassName, true);
#pragma warning disable CS0618
    else if (AutoDownUnreachableAfter.HasValue)
#pragma warning restore CS0618
        DowningProviderType = typeof(AutoDowning);
    else
        DowningProviderType = typeof(NoDowning);

    RunCoordinatedShutdownWhenDown = clusterConfig.GetBoolean("run-coordinated-shutdown-when-down", false);

    TimeSpan GetWeaklyUpDuration()
    {
        var cKey = "allow-weakly-up-members";
        switch (clusterConfig.GetString(cKey, string.Empty)
            .ToLowerInvariant())
        {
            case "off":
                return TimeSpan.Zero;
            case "on":
                return TimeSpan.FromSeconds(7); // for backwards compatibility when it wasn't a duration
            default:
                var val = clusterConfig.GetTimeSpan(cKey, TimeSpan.FromSeconds(7));
                if (!val > TimeSpan.Zero)
                    throw new ConfigurationException($"Valid settings for [akka.cluster.{cKey}] are 'off', 'on', or a timespan greater than 0s. Received [{val}]");
                return val;
        }
    }

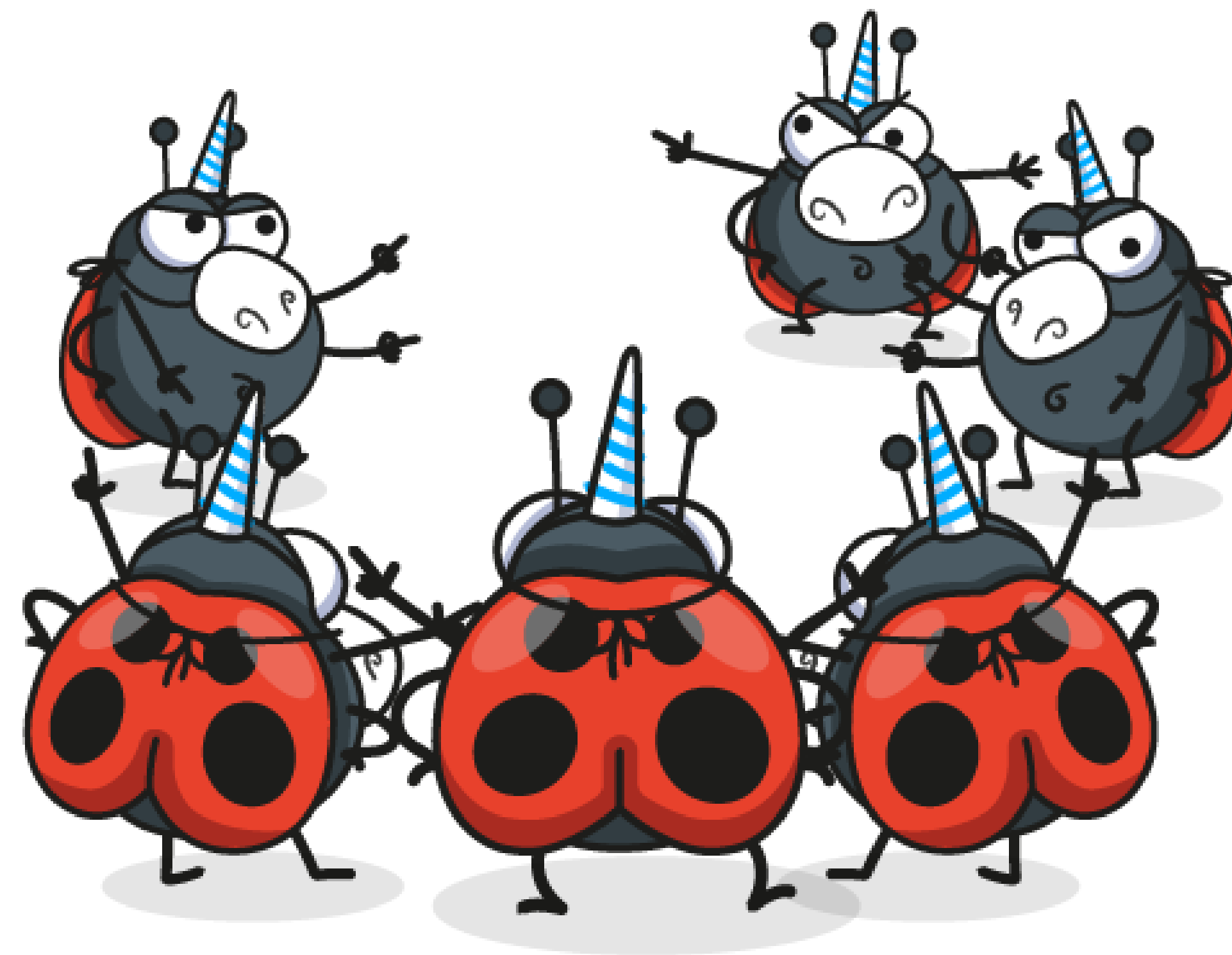
    WeaklyUpAfter = GetWeaklyUpDuration();

    UseLegacyHeartbeatMessage = clusterConfig.GetBoolean("use-legacy-heartbeat-message", false);
}
```

```
MinNrOfMembersOfRole =
clusterConfig
    .GetConfig("role")
    .Root.GetObject()
    .Items
    .ToImmutableDictionary(
        kv => kv.Key,
        kv => kv.Value.GetObject()?.GetKey(...).GetInt()
    );
```

# МЕНЯЕМСЯ

Сложные switch-case



```

@Override
public void setDatasource(final CollectionDatasource datasource) {
    ....
    collectionChangeListener = e -> {
        switch (e.getOperation()) {
            case CLEAR:
            case REFRESH:
                fieldDatasources.clear();
                break;

            case UPDATE:
            case REMOVE:
                for (Object entity : e.getItems()) {
                    fieldDatasources.remove(entity);
                }
                break;
        }
    };
    ....
}

```



```

@Override
public void setDatasource(final CollectionDatasource datasource) {
    ....
    collectionChangeListener = e -> {
        switch (e.getOperation()) {
            case CLEAR:
            case REFRESH:
                fieldDatasources.clear();
                break;

            case UPDATE:
            case REMOVE:
                for (Object entity : e.getItems()) {
                    fieldDatasources.remove(entity);
                }
                break;
        }
    };
    ....
}

```



```

enum Operation {
    REFRESH,
    CLEAR,
    ADD,
    REMOVE,
    UPDATE
}

```

```

@Override
public void setDatasource(final CollectionDatasource datasource) {
    ....
    collectionChangeListener = e -> {
        switch (e.getOperation()) {
            case CLEAR:
            case REFRESH:
                fieldDatasources.clear();
                break;

            case UPDATE:
            case REMOVE:
                for (Object entity : e.getItems()) {
                    fieldDatasources.remove(entity);
                }
                break;
        }
    };
    ....
}

```



```

enum Operation {
    REFRESH,
    ✓ CLEAR,
    ADD,
    REMOVE,
    UPDATE
}

```



```

@Override
public void setDatasource(final CollectionDatasource datasource) {
    ....
    collectionChangeListener = e -> {
        switch (e.getOperation()) {
            case CLEAR:
                case REFRESH:
                    fieldDatasources.clear();
                    break;

            case UPDATE:
            case REMOVE:
                for (Object entity : e.getItems()) {
                    fieldDatasources.remove(entity);
                }
                break;
        }
    };
    ....
}

```



```

enum Operation {
    ✓ REFRESH,
    ✓ CLEAR,
    ADD,
    REMOVE,
    UPDATE
}

```

```

@Override
public void setDatasource(final CollectionDatasource datasource) {
    ....
    collectionChangeListener = e -> {
        switch (e.getOperation()) {
            case CLEAR:
            case REFRESH:
                fieldDatasources.clear();
                break;

case UPDATE:
            case REMOVE:
                for (Object entity : e.getItems()) {
                    fieldDatasources.remove(entity);
                }
                break;
        }
    };
    ....
}

```



```

enum Operation {
    ✓ REFRESH,
    ✓ CLEAR,
    ADD,
    REMOVE,
    ✓ UPDATE
}

```

```

@Override
public void setDatasource(final CollectionDatasource datasource) {
    ....
    collectionChangeListener = e -> {
        switch (e.getOperation()) {
            case CLEAR:
            case REFRESH:
                fieldDatasources.clear();
                break;

            case UPDATE:
case REMOVE:
                for (Object entity : e.getItems()) {
                    fieldDatasources.remove(entity);
                }
                break;
        }
    };
    ....
}

```



```

enum Operation {
    ✓ REFRESH,
    ✓ CLEAR,
    ADD,
    ✓ REMOVE,
    ✓ UPDATE
}

```

```

@Override
public void setDatasource(final CollectionDatasource datasource) {
    ....
    collectionChangeListener = e -> {
        switch (e.getOperation()) {
            case CLEAR:
            case REFRESH:
                fieldDatasources.clear();
                break;

            case UPDATE:
            case REMOVE:
                for (Object entity : e.getItems()) {
                    fieldDatasources.remove(entity);
                }
                break;
        }
    };
    ....
}

```

```

enum Operation {
    ✓ REFRESH,
    ✓ CLEAR,
    ADD,
    ✓ REMOVE,
    ✓ UPDATE
}

```

**V6002** The switch statement does not cover all values of the 'Operation' enum: ADD.



# Изменили enum





Изменили enum



Нужно изменить все места в  
коде, где он перебирается

```

switch (tc)
{
    case TypeCode.Empty: break;
    case TypeCode.Object:
        variant = ComVariant.CreateRaw(...); break;
    case TypeCode.DBNull:
        variant = ComVariant.Null; break;
    case TypeCode.Boolean:
        variant = ComVariant.Create<bool>(...); break;
    case TypeCode.Char:
        variant = ComVariant.Create<ushort>(value.ToChar(ci)); break;
    case TypeCode.SByte:
        variant = ComVariant.Create<sbyte>(value.ToSByte(ci)); break;
    case TypeCode.Byte:
        variant = ComVariant.Create<byte>(value.ToByte(ci)); break;
    case TypeCode.Int16:
        variant = ComVariant.Create(value.ToInt16(ci)); break;
    case TypeCode.UInt16:
        variant = ComVariant.Create(value.ToUInt16(ci)); break;
    case TypeCode.Int32:
        variant = ComVariant.Create(value.ToInt32(ci)); break;
    case TypeCode.UInt32:
        variant = ComVariant.Create(value.ToUInt32(ci)); break;
    case TypeCode.Int64:
        variant = ComVariant.Create(value.ToInt64(ci)); break;
    case TypeCode.UInt64:
        variant = ComVariant.Create(value.ToUInt64(ci)); break;
    case TypeCode.Single:
        variant = ComVariant.Create(value.ToSingle(ci)); break;
    case TypeCode.Double:
        variant = ComVariant.Create(value.ToDouble(ci)); break;
    case TypeCode.Decimal:
        variant = ComVariant.Create(value.ToDecimal(ci)); break;
    case TypeCode.DateTime:
        variant = ComVariant.Create(value.ToDateTime(ci)); break;
    case TypeCode.String:
        variant = ComVariant.Create(...); break;

    default:
        throw new NotSupportedException();
}

```



```

switch (tc)
{
    case TypeCode.Empty: break;
    case TypeCode.Object:
        variant = ComVariant.CreateRaw(...); break;
    case TypeCode.DBNull:
        variant = ComVariant.Null; break;
    case TypeCode.Boolean:
        variant = ComVariant.Create<bool>(....); break;
    case TypeCode.Char:
        variant = ComVariant.Create<ushort>(value.ToChar(ci)); break;
    case TypeCode.SByte:
        variant = ComVariant.Create<sbyte>(value.ToSByte(ci)); break;
    case TypeCode.Byte:
        variant = ComVariant.Create<byte>(value.ToByte(ci)); break;
    case TypeCode.Int16:
        variant = ComVariant.Create(value.ToInt16(ci)); break;
    case TypeCode.UInt16:
        variant = ComVariant.Create(value.ToUInt16(ci)); break;
    case TypeCode.Int32:
        variant = ComVariant.Create(value.ToInt32(ci)); break;
    case TypeCode.UInt32:
        variant = ComVariant.Create(value.ToUInt32(ci)); break;
    case TypeCode.Int64:
        variant = ComVariant.Create(value.ToInt64(ci)); break;
    case TypeCode.UInt64:
        variant = ComVariant.Create(value.ToUInt64(ci)); break;
    case TypeCode.Single:
        variant = ComVariant.Create(value.ToSingle(ci)); break;
    case TypeCode.Double:
        variant = ComVariant.Create(value.ToDouble(ci)); break;
    case TypeCode.Decimal:
        variant = ComVariant.Create(value.ToDecimal(ci)); break;
    case TypeCode.DateTime:
        variant = ComVariant.Create(value.ToDateTime(ci)); break;
    case TypeCode.String:
        variant = ComVariant.Create(...); break;

    default:
        throw new NotSupportedException();
}

```





```
switch (tc)
{
    case TypeCode.Empty: break;
    case TypeCode.Object:
        variant = ComVariant.CreateRaw(...); break;
    case TypeCode.DBNull:
        variant = ComVariant.Null; break;
    case TypeCode.Boolean:
        variant = ComVariant.Create<bool>(....); break;
    case TypeCode.Char:
        variant = ComVariant.Create<ushort>(value.ToChar(ci)); break;
    case TypeCode.SByte:
        variant = ComVariant.Create<sbyte>(value.ToSByte(ci)); break;
    case TypeCode.Byte:
        variant = ComVariant.Create<byte>(value.ToByte(ci)); break;
    case TypeCode.Int16:
        variant = ComVariant.Create(value.ToInt16(ci)); break;
    case TypeCode.UInt16:
        variant = ComVariant.Create(value.ToUInt16(ci)); break;
    case TypeCode.Int32:
        variant = ComVariant.Create(value.ToInt32(ci)); break;
    case TypeCode.UInt32:
        variant = ComVariant.Create(value.ToUInt32(ci)); break;
    case TypeCode.Int64:
        variant = ComVariant.Create(value.ToInt64(ci)); break;
    case TypeCode.UInt64:
        variant = ComVariant.Create(value.ToUInt64(ci)); break;
    case TypeCode.Single:
        variant = ComVariant.Create(value.ToSingle(ci)); break;
    case TypeCode.Double:
        variant = ComVariant.Create(value.ToDouble(ci)); break;
    case TypeCode.Decimal:
        variant = ComVariant.Create(value.ToDecimal(ci)); break;
    case TypeCode.DateTime:
        variant = ComVariant.Create(value.ToDateTime(ci)); break;
    case TypeCode.String:
        variant = ComVariant.Create(...); break;

    default:
        throw new NotSupportedException();
}
```

```
case TypeCode.Int64:
    variant = ComVariant.Create(value.ToInt64(ci));
    break;
case TypeCode.UInt64:
    variant = ComVariant.Create(value.ToUInt64(ci));
    break;
```

```

switch (tc)
{
    case TypeCode.Empty: break;
    case TypeCode.Object:
        variant = ComVariant.CreateRaw(...); break;
    case TypeCode.DBNull:
        variant = ComVariant.Null; break;
    case TypeCode.Boolean:
        variant = ComVariant.Create<bool>(....); break;
    case TypeCode.Char:
        variant = ComVariant.Create<ushort>(value.ToChar(ci)); break;
    case TypeCode.SByte:
        variant = ComVariant.Create<sbyte>(value.ToSByte(ci)); break;
    case TypeCode.Byte:
        variant = ComVariant.Create<byte>(value.ToByte(ci)); break;
    case TypeCode.Int16:
        variant = ComVariant.Create(value.ToInt16(ci)); break;
    case TypeCode.UInt16:
        variant = ComVariant.Create(value.ToUInt16(ci)); break;
    case TypeCode.Int32:
        variant = ComVariant.Create(value.ToInt32(ci)); break;
    case TypeCode.UInt32:
        variant = ComVariant.Create(value.ToUInt32(ci)); break;
    case TypeCode.Int64:
        variant = ComVariant.Create(value.ToInt64(ci)); break;
    case TypeCode.UInt64:
        variant = ComVariant.Create(value.ToInt64(ci)); break;
    case TypeCode.Single:
        variant = ComVariant.Create(value.ToSingle(ci)); break;
    case TypeCode.Double:
        variant = ComVariant.Create(value.ToDouble(ci)); break;
    case TypeCode.Decimal:
        variant = ComVariant.Create(value.ToDecimal(ci)); break;
    case TypeCode.DateTime:
        variant = ComVariant.Create(value.ToDateTime(ci)); break;
    case TypeCode.String:
        variant = ComVariant.Create(...); break;

    default:
        throw new NotSupportedException();
}

```

```
case TypeCode.Int64:
```

```
    variant = ComVariant.Create(value.ToInt64(ci));
    break;
```

```
case TypeCode.UInt64:
```

```
    variant = ComVariant.Create(value.ToInt64(ci));
    break;
```

**V3139** Two or more case-branches perform the same actions.



# ЧТО МЕРТВО, УМЕРЕТЬ НЕ МОЖЕТ

Мёртвый код





```
template<typename Scalar>
vector<Scalar> operator*(
    double scalar,
    const vector<Scalar>& v
) {
    vector<Scalar> scaledVector(v);
    scaledVector *= scalar;
    return v;
}
```

```
template<typename Scalar>
vector<Scalar> operator*(
    double scalar,
    const vector<Scalar>& v
) {
    vector<Scalar> scaledVector(v);
    scaledVector *= scalar;
    return v;
}
```

**V1001** The 'scaledVector' variable is assigned but is not used by the end of the function.

«Мёртвым называется код, не выполняемый в ходе работы программы. Он содержится в теле команды `if`, проверяющей невозможное условие. Он содержится в секции `catch` для блока `try`, никогда не инициирующего исключения. Он содержится в маленьких вспомогательных методах, которые никогда не вызываются, или в никогда не встречающихся условиях `switch/case`.»

## РОБЕРТ МАРТИН

«Clean Code: A Handbook of Agile Software Craftsmanship»



```

@Override
public NextAction handleRead(FilterChainContext context) throws IOException {
    ....
    do {
        savedReadIndex = frame.readerIndex();
        ....
        if (msg == Codec2.DecodeResult.NEED_MORE_INPUT) {
            ....
            return context.getStopAction();
        } else {
            if (savedReadIndex == frame.readerIndex()) {
                ....
                throw new IOException("Decode without read data.");
            }
            if (msg != null) {
                ....
                return context.getInvokeAction();
            } else {
                ....
                return context.getInvokeAction();
            }
        }
    } while (frame.readable());
    ....
}

```



```

@Override
public NextAction handleRead(FilterChainContext context) throws IOException {
    ....
    do {
        savedReadIndex = frame.readerIndex();
        ....
        if (msg == Codec2.DecodeResult.NEED_MORE_INPUT) {
            ....
            return context.getStopAction();
        } else {
            if (savedReadIndex == frame.readerIndex()) {
                ....
                throw new IOException("Decode without read data.");
            }
            if (msg != null) {
                ....
                return context.getInvokeAction();
            } else {
                ....
                return context.getInvokeAction();
            }
        }
    }
    while (frame.readable());
    ....
}

```



```

@Override
public NextAction handleRead(FilterChainContext context) throws IOException {
    ....
    do {
        savedReadIndex = frame.readerIndex();
        ....
        if (msg == Codec2.DecodeResult.NEED_MORE_INPUT) {
            ....
            return context.getStopAction();
        } else {
            if (savedReadIndex == frame.readerIndex()) {
                ....
                throw new IOException("Decode without read data.");
            }
            if (msg != null) {
                ....
                return context.getInvokeAction();
            } else {
                ....
                return context.getInvokeAction();
            }
        }
    } while (frame.readable());
    ....
}

```



```

@Override
public NextAction handleRead(FilterChainContext context) throws IOException {
    ....
    do {
        savedReadIndex = frame.readerIndex();
        ....
        if (msg == Codec2.DecodeResult.NEED_MORE_INPUT) {
            ....
            return context.getStopAction();
        } else {
            if (savedReadIndex == frame.readerIndex()) {
                ....
                throw new IOException("Decode without read data.");
            }
            if (msg != null) {
                ....
                return context.getInvokeAction();
            } else {
                ....
                return context.getInvokeAction();
            }
        }
    } while (frame.readable());
    ....
}

```





```

@Override
public NextAction handleRead(FilterChainContext context) throws IOException {
    ....
    do {
        savedReadIndex = frame.readerIndex();
        ....
        if (msg == Codec2.DecodeResult.NEED_MORE_INPUT) {
            ....
            return context.getStopAction();
        } else {
            if (savedReadIndex == frame.readerIndex()) {
                ....
                throw new IOException("Decode without read data.");
            }
            if (msg != null) {
                ....
                return context.getInvokeAction();
            } else {
                ....
                return context.getInvokeAction();
            }
        }
    } while (frame.readable());
    ....
}

```



```

@Override
public NextAction handleRead(FilterChainContext context) throws IOException {
    ....
    do {
        savedReadIndex = frame.readerIndex();
        ....
        if (msg == Codec2.DecodeResult.NEED_MORE_INPUT) {
            ....
            return context.getStopAction();
        } else {
            if (savedReadIndex == frame.readerIndex()) {
                ....
                throw new IOException("Decode without read data.");
            }
            if (msg != null) {
                ....
                return context.getInvokeAction();
            } else {
                ....
                return context.getInvokeAction();
            }
        }
    } while (frame.readable());
    ....
}

```



```

@Override
public NextAction handleRead(FilterChainContext context) throws IOException {
    ....
    do {
        savedReadIndex = frame.readerIndex();
        ....
        if (msg == Codec2.DecodeResult.NEED_MORE_INPUT) {
            ....
            return context.getStopAction();
        } else {
            if (savedReadIndex == frame.readerIndex()) {
                ....
                throw new IOException("Decode without read data.");
            }
            if (msg != null) {
                ....
                return context.getInvokeAction();
            } else {
                ....
                return context.getInvokeAction();
            }
        }
    } while (frame.readable());
    ....
}

```



```

@Override
public NextAction handleRead(FilterChainContext context) throws IOException {
    ....
    do {
        savedReadIndex = frame.readerIndex();
        ....
        if (msg == Codec2.DecodeResult.NEED_MORE_INPUT) {
            ....
            return context.getStopAction();
        } else {
            if (savedReadIndex == frame.readerIndex()) {
                ....
                throw new IOException("Decode without read data.");
            }
            if (msg != null) {
                ....
                return context.getInvokeAction();
            } else {
                ....
                return context.getInvokeAction();
            }
        }
    } while (frame.readable());
    ....
}

```



**V6019** Unreachable code detected. It is possible that an error is present.

# ...ВСЁ ХОДИТ ПО ЦЕПИ КРУГОМ

Цепочки вызовов



....

```
w.NumChannels = AUD_BUF->audinfo().channels();  
w.SampleRate = AUD_BUF->audinfo().sample_rate();  
w.ByteRate = AUD_BUF->audinfo().byte_rate();  
w.BlockAlign = AUD_BUF->audinfo().block_align();  
w.BitsPerSample = AUD_BUF->audinfo().bits();
```

....



```
....  
w.NumChannels = AUD_BUF->audinfo().channels();  
w.SampleRate = AUD_BUF->audinfo().sample_rate();  
w.ByteRate = AUD_BUF->audinfo().byte_rate();  
w.BlockAlign = AUD_BUF->audinfo().block_align();  
w.BitsPerSample = AUD_BUF->audinfo().bits();  
....
```

**V807** Decreased performance. Consider creating a reference to avoid using the 'AUD\_BUF->audinfo()' expression repeatedly.



```
....  
w.NumChannels = AUD_BUF->audinfo().channels();  
w.SampleRate = AUD_BUF->audinfo().sample_rate();  
w.ByteRate = AUD_BUF->audinfo().byte_rate();  
w.BlockAlign = AUD_BUF->audinfo().block_align();  
w.BitsPerSample = AUD_BUF->audinfo().bits();  
....
```

**V807** Decreased performance. Consider creating a reference to avoid using the 'AUD\_BUF->audinfo()' expression repeatedly.

```
....  
auto audinfo = AUD_BUF->audinfo();  
  
w.NumChannels = audinfo.channels();  
w.SampleRate = audinfo.sample_rate();  
w.ByteRate = audinfo.byte_rate();  
w.BlockAlign = audinfo.block_align();  
w.BitsPerSample = audinfo.bits();  
....
```

V807 Decreased performance. Consider creating a reference to avoid using the 'AUD\_BUF->audinfo()' expression repeatedly.

# ПОЧЕМУ ЭТО ПЛОХО

- при изменении промежуточных связей нам потребуется изменить и клиент;



# ПОЧЕМУ ЭТО ПЛОХО

- при изменении промежуточных связей нам потребуется изменить и клиент;
- цепочка вызовов требует больше времени, чтобы её прочитать и понять;





# ПОЧЕМУ ЭТО ПЛОХО

- при изменении промежуточных связей нам потребуется изменить и клиент;
- цепочка вызовов требует больше времени, чтобы её прочитать и понять;
- лишняя трата производительности.



# ПОЧЕМУ ЭТО ПЛОХО

- при изменении промежуточных связей нам потребуется изменить и клиент;
- цепочка вызовов требует больше времени, чтобы её прочитать и понять;
- лишняя трата производительности.

Поправить можно рефакторингом.



```
for (size_t keypointNum = 0u;
    keypointNum < numberOfKeypoints; ++keypointNum)
{
    os << obj.__features.__objectKeypoints[keypointNum].pt.x << ' ';
    os << obj.__features.__objectKeypoints[keypointNum].pt.y << ' ';
    os << obj.__features.__objectKeypoints[keypointNum].size << ' ';
    os << obj.__features.__objectKeypoints[keypointNum].response << ' ';
    os << obj.__features.__objectKeypoints[keypointNum].angle << ' ';
    os << obj.__features.__objectKeypoints[keypointNum].octave << ' ';
    os << obj.__features.__objectKeypoints[keypointNum].class_id << '\n';
}
```





```
for (size_t keypointNum = 0u;
    keypointNum < numberOfKeypoints; ++keypointNum)
{
    os << obj.__features.__objectKeypoints[keypointNum].pt.x << ' ';
    os << obj.__features.__objectKeypoints[keypointNum].pt.y << ' ';
    os << obj.__features.__objectKeypoints[keypointNum].size << ' ';
    os << obj.__features.__objectKeypoints[keypointNum].response << ' ';
    os << obj.__features.__objectKeypoints[keypointNum].angle << ' ';
    os << obj.__features.__objectKeypoints[keypointNum].octave << ' ';
    os << obj.__features.__objectKeypoints[keypointNum].class_id << '\n';
}
```

**V807** Decreased performance. Consider creating a reference to avoid using the same expression repeatedly.

```
for (...) {  
    auto &keypoint = obj.__features.__objectKeypoints[keypointNum];  
    os << keypoint.pt.x << ' '  
        << keypoint.pt.y << ' '  
        << keypoint.size << ' '  
        << keypoint.response << ' '  
        << keypoint.angle << ' '  
        << keypoint.octave << ' '  
        << keypoint.class_id << '\\n';  
}
```



~~V807 Decreased performance. Consider creating a reference to avoid using the same expression repeatedly.~~

# ПОЧЕМУ ЭТО ПЛОХО

- при изменении промежуточных связей нам потребуется изменить и клиент;
- цепочка вызовов требует больше времени, чтобы её прочитать и понять;
- лишняя трата производительности.

Поправить можно рефакторингом.

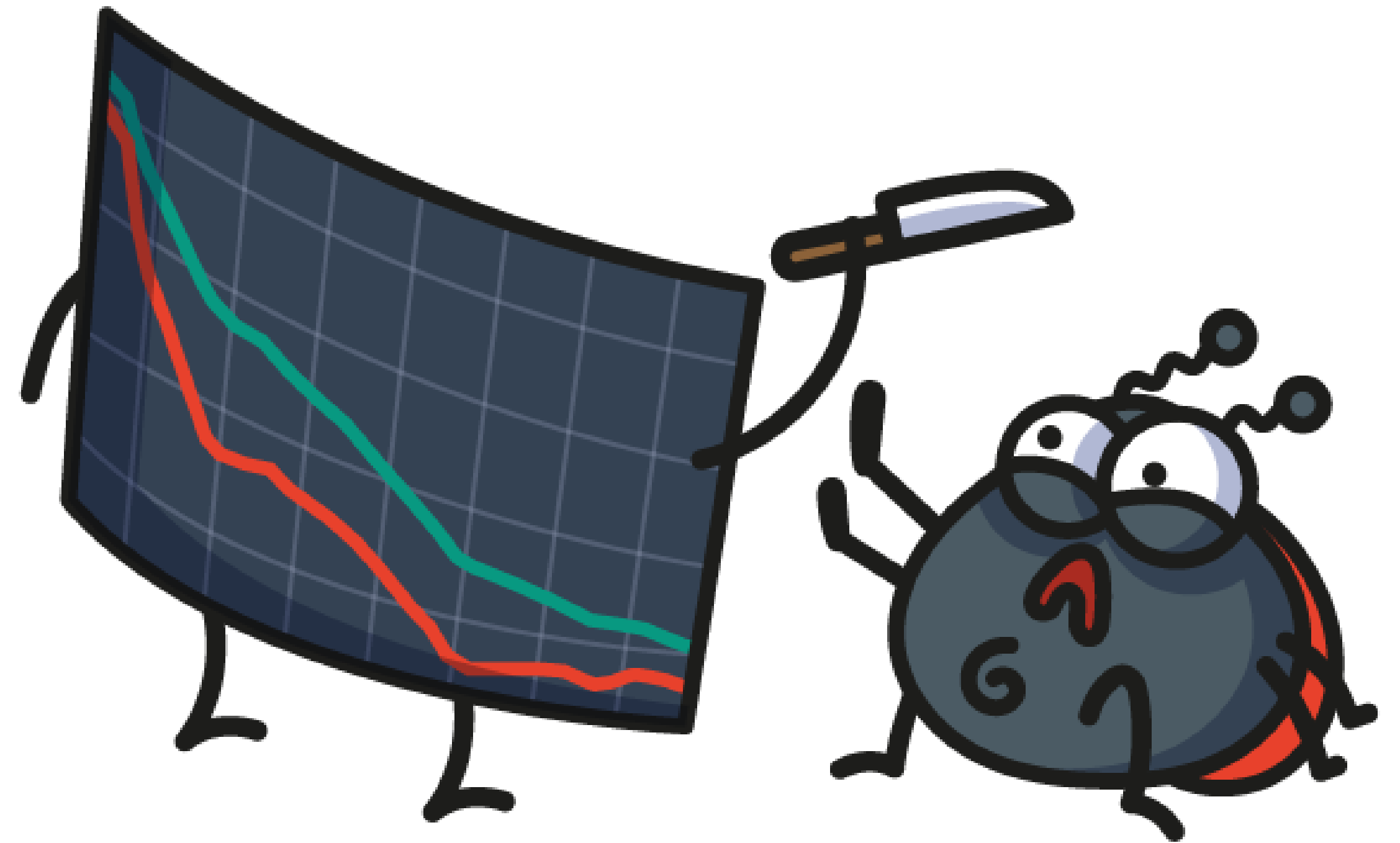
В сложных случаях:

- [Hide Delegate](#)
- [Extract Method](#)
- [Move Method.](#)



# УБИЙСТВЕННЫЙ ПЕРФОМАНС

Аргументы против



Эти правила конкретизируют, как написать код так, чтобы он считался "чистым". Но вот вопрос: если мы напишем код, следуя всем этим правилам, что у него будет с производительностью?

## КЕЙСИ МУРАТОРИ

Lead programmer at Molly Rocket, Inc.,  
автор видео «"Clean" Code, Horrible  
Performance»



# ДИСКУССИЯ

- «Почему ты нас не предупредил?»
- Инженеры vs GitHub
- Читаемость и тесты
- Классы против if/switch



## Статья

Чистый код — дар или проклятие?  
Акт I. Конфронтация



# SUMMARY

Подводим итоги





# ИТОГИ

- ЧИСТЫЙ КОД ЭТО КРУТО



# ИТОГИ

- Чистый код это круто
- Недостаточная чистота кода влияет на его работоспособность



# ИТОГИ

- Чистый код это круто
- Недостаточная чистота кода влияет на его работоспособность
- Во всём нужно знать меру :)



# Q/A



@feelindex



filatov@pvs-studio.ru

