Smart Code editors with JavaFX (and e4)

Tom Schindl <tom.schindl@bestsolution.at>

Twitter: @tomsontom
Blog: http://tomsondev.bestsolution.at
Website: http://www.bestsolution.at

About Me

- CTO BestSolution.at Systemhaus GmbH
- Eclipse Committer
  - e4
  - Platform
  - EMF
- Project lead
  - e(fx)clipse
- Twitter: @tomsontom
- Blog: tomsondev.bestsolution.at
- Cooperate: http://bestsolution.at

Today

Today

- Current situation
  - Eclipse 4 Application Platform does not support editors
  - Eclipse IDE / Eclipse 3.x RCP is still required to get an editor framework

Today

- Current situation
  - Eclipse 4 Application Platform does not support editors
  - Eclipse IDE / Eclipse 3.x RCP is still required to get an editor framework
- Future situation
  - Eclipse 4 should support editors natively
  - Loosely coupled components / services who work in Eclipse4, plain Java, jigsaw world

Eclipse 3.x Codeeditors

- Should we directly port the 3.x code-editor stuff to Eclipse 4?

Eclipse 3.x Codeeditors

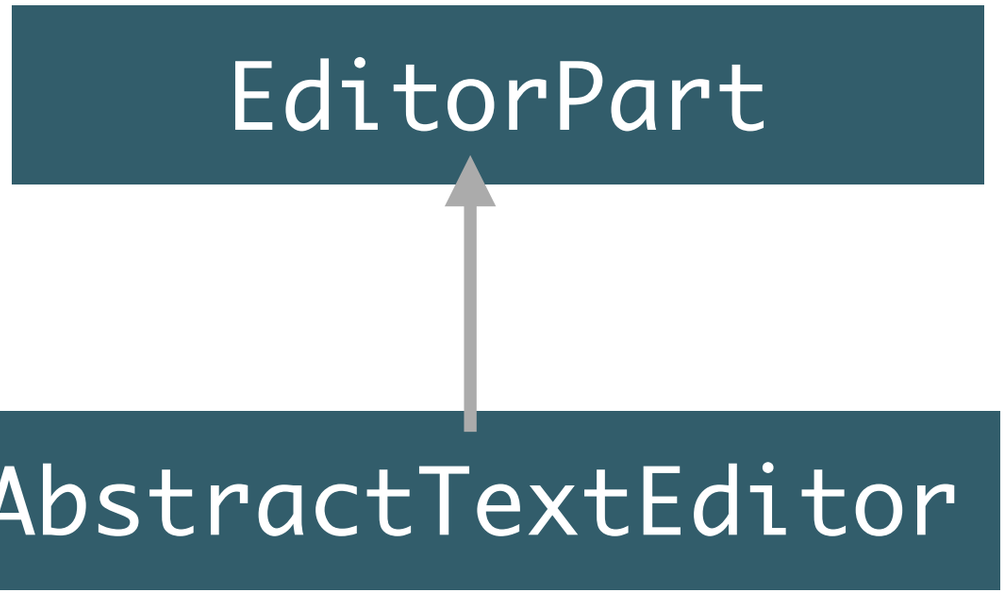
- Should we directly port the 3.x code-editor stuff to Eclipse 4?

EditorPart

Eclipse 3.x Codeeditors

BestSolution

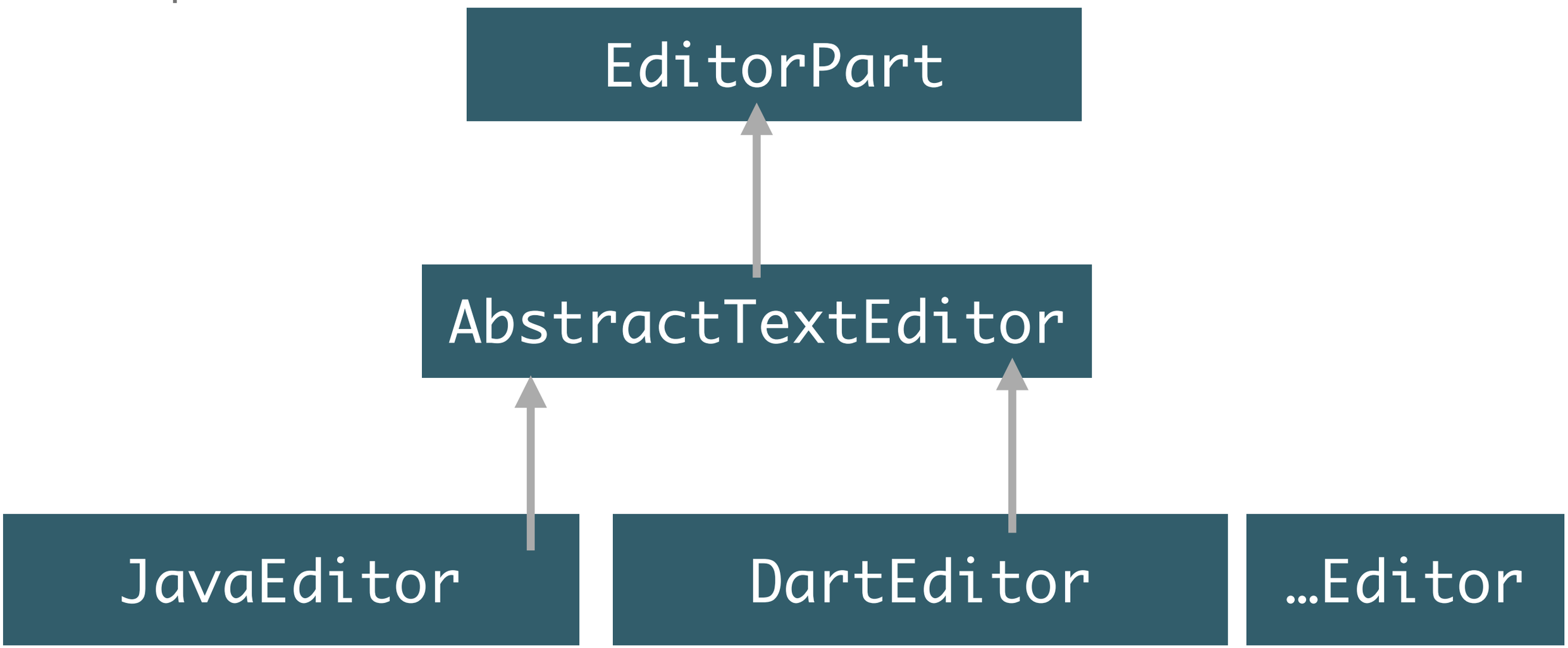
- Should we directly port the 3.x code-editor stuff to Eclipse 4?



Eclipse 3.x Codeeditors

BestSolution

- Should we directly port the 3.x code-editor stuff to Eclipse 4?



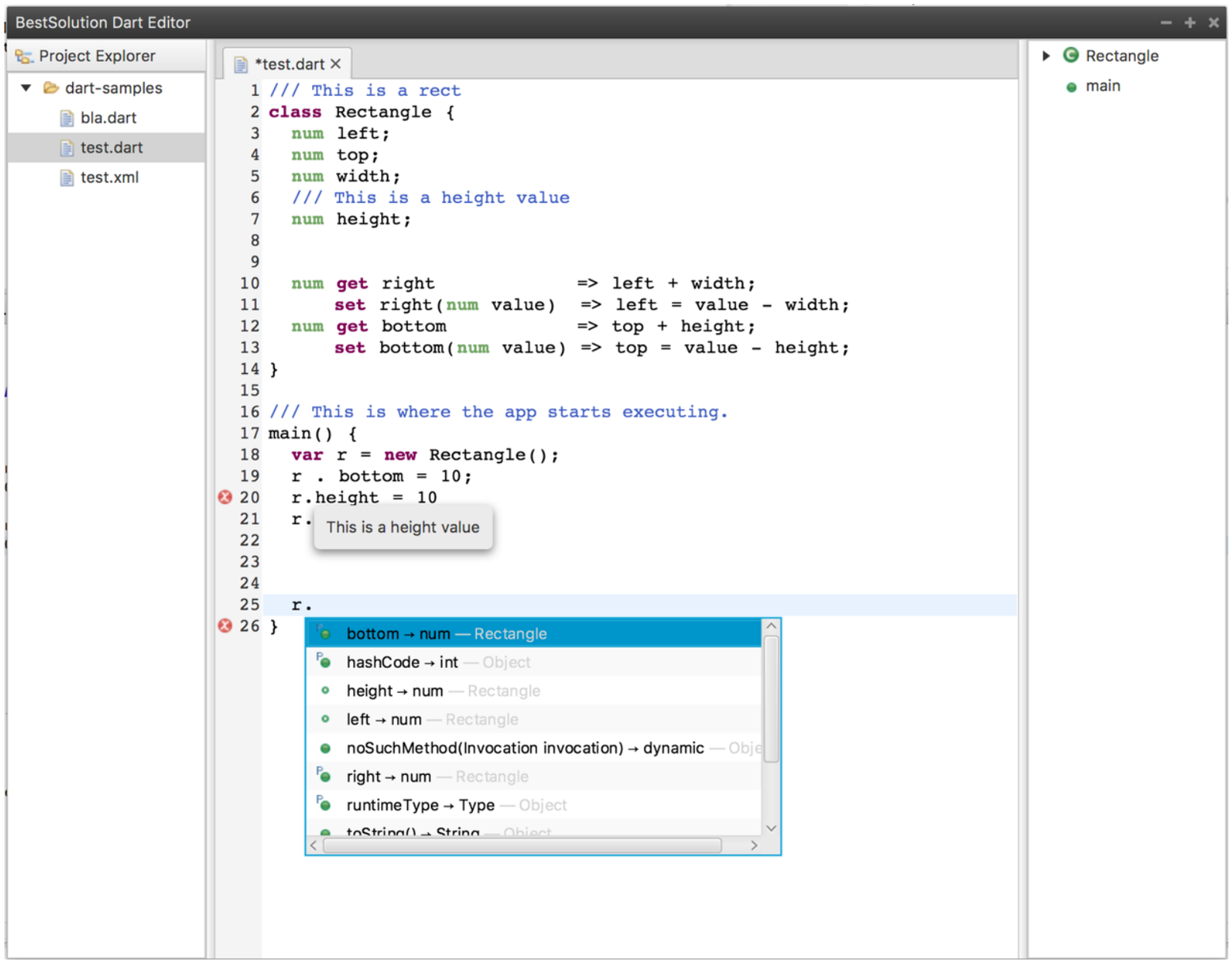
Codeeditors the e4 way

- Editors in a DI/Service env

Codeeditors the e4 way

BestSolution

- Editors in a DI/Service env

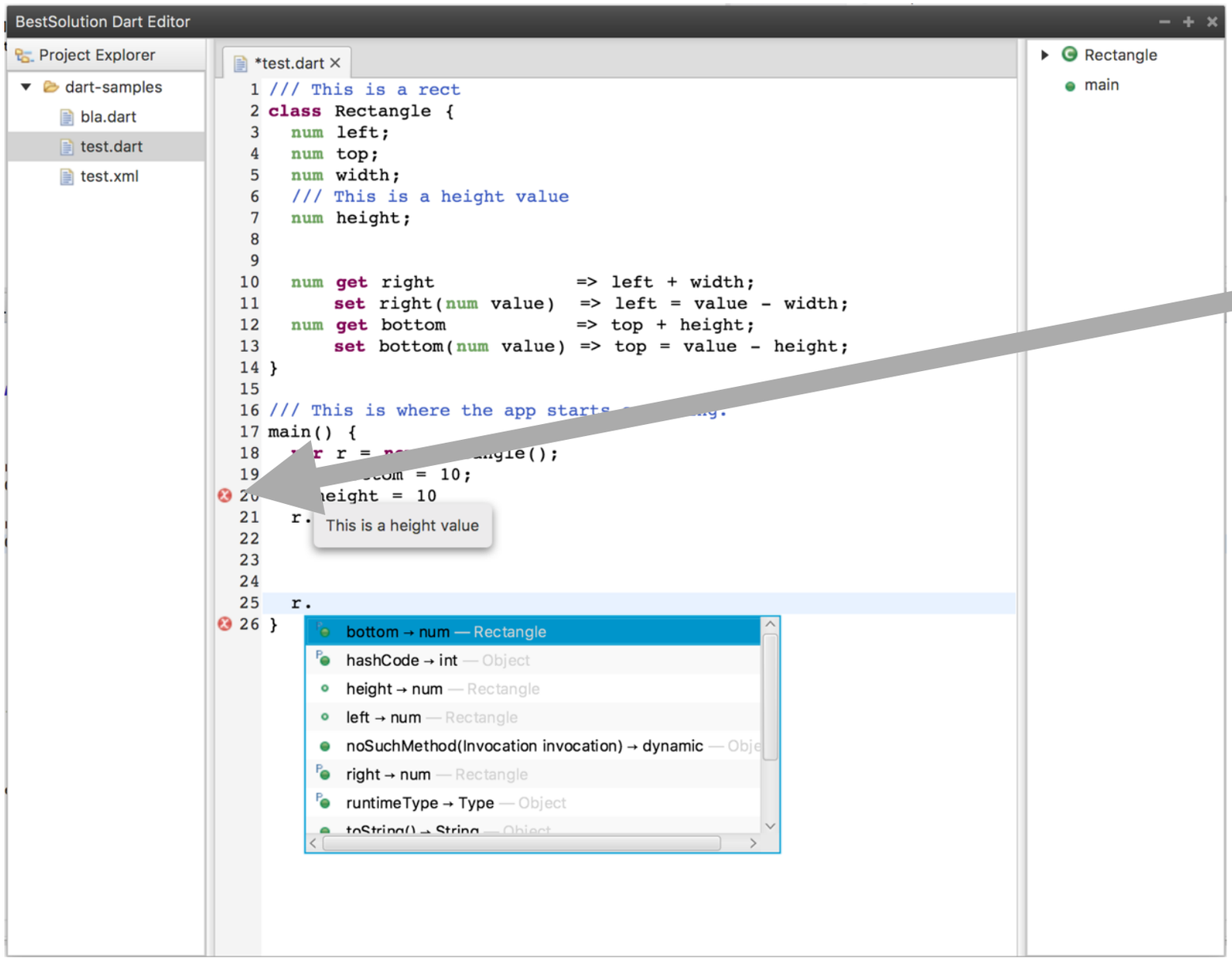


Syntax-Highlighting

Codeeditors the e4 way

BestSolution

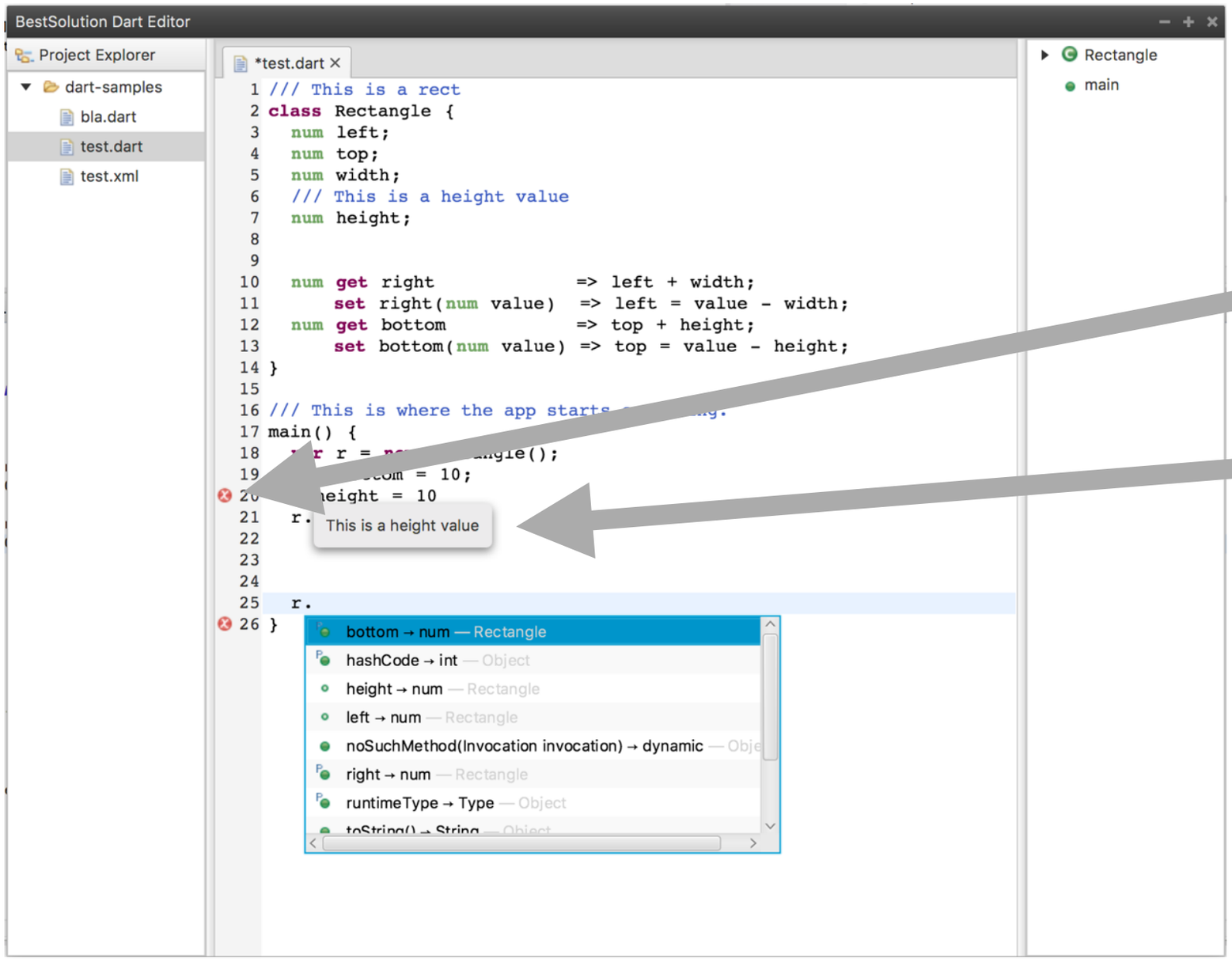
- Editors in a DI/Service env



Codeeditors the e4 way

BestSolution

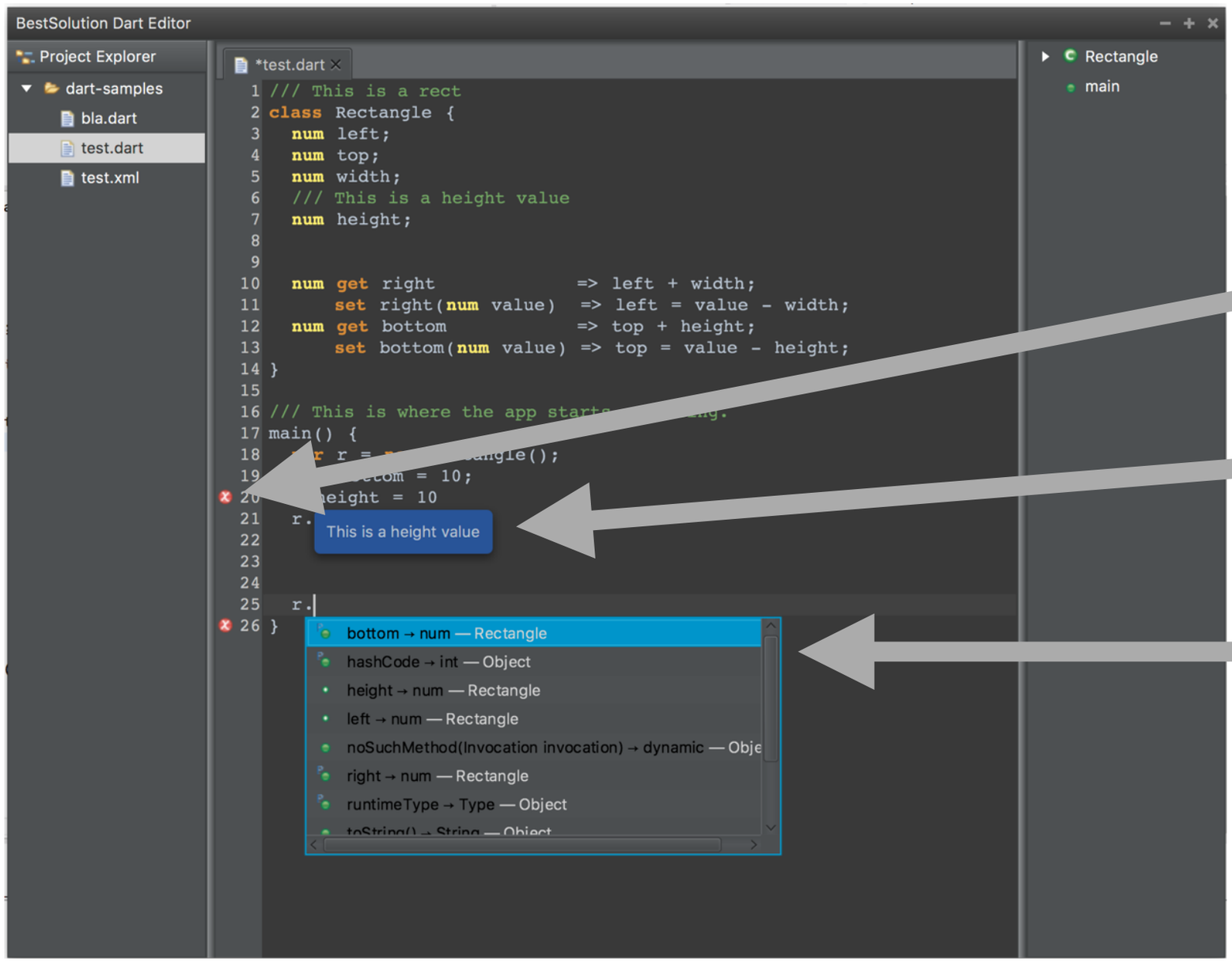
- Editors in a DI/Service env



Codeeditors the e4 way

BestSolution

- Editors in a DI/Service env



Syntax-Highlighting

Error Marker

Documentation Info

Autocomplete

IEditorInput replacement

- Basic API is named `Input<O>`

IEditorInput replacement

- Basic API is named `Input<O>`

```
public interface Input<O> {
  public void dispose();

  public O getData();

  public void setData(O data);

  public void persist();
}
```

IEditorInput replacement

- Basic API is named `Input<O>`

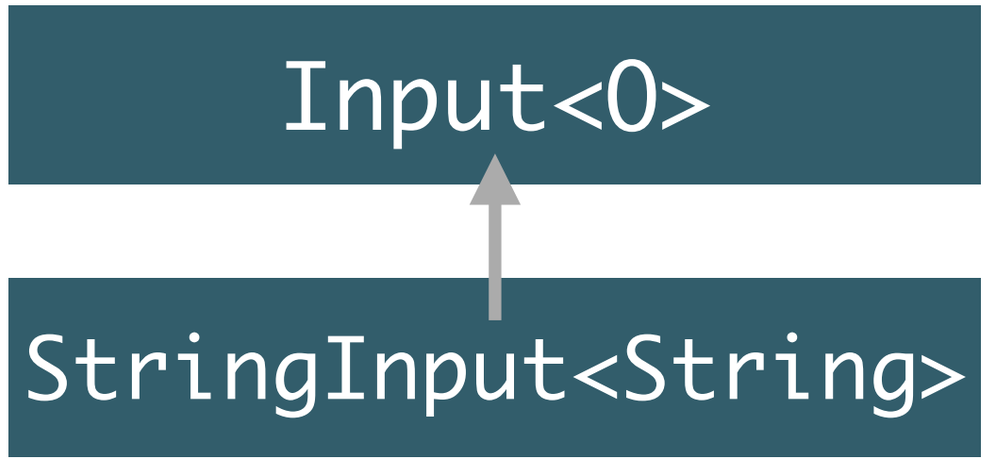
```
public interface Input<O> {
  public void dispose();

  public O getData();

  public void setData(O data);

  public void persist();
}
```

Input<O>

IEditorInput replacement

- Basic API is named `Input<O>`

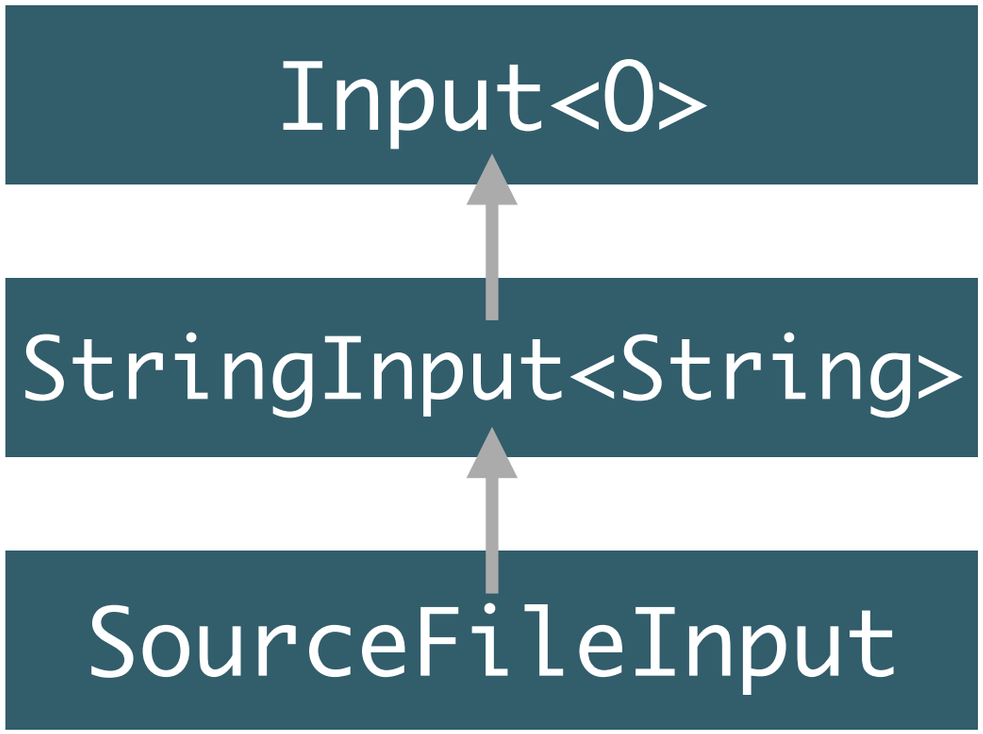
```
public interface Input<O> {
  public void dispose();

  public O getData();

  public void setData(O data);

  public void persist();
}
```



IEditorInput replacement

- Basic API is named `Input<O>`

```
public interface Input<O> {
  public void dispose();

  public O getData();

  public void setData(O data);

  public void persist();
}
```

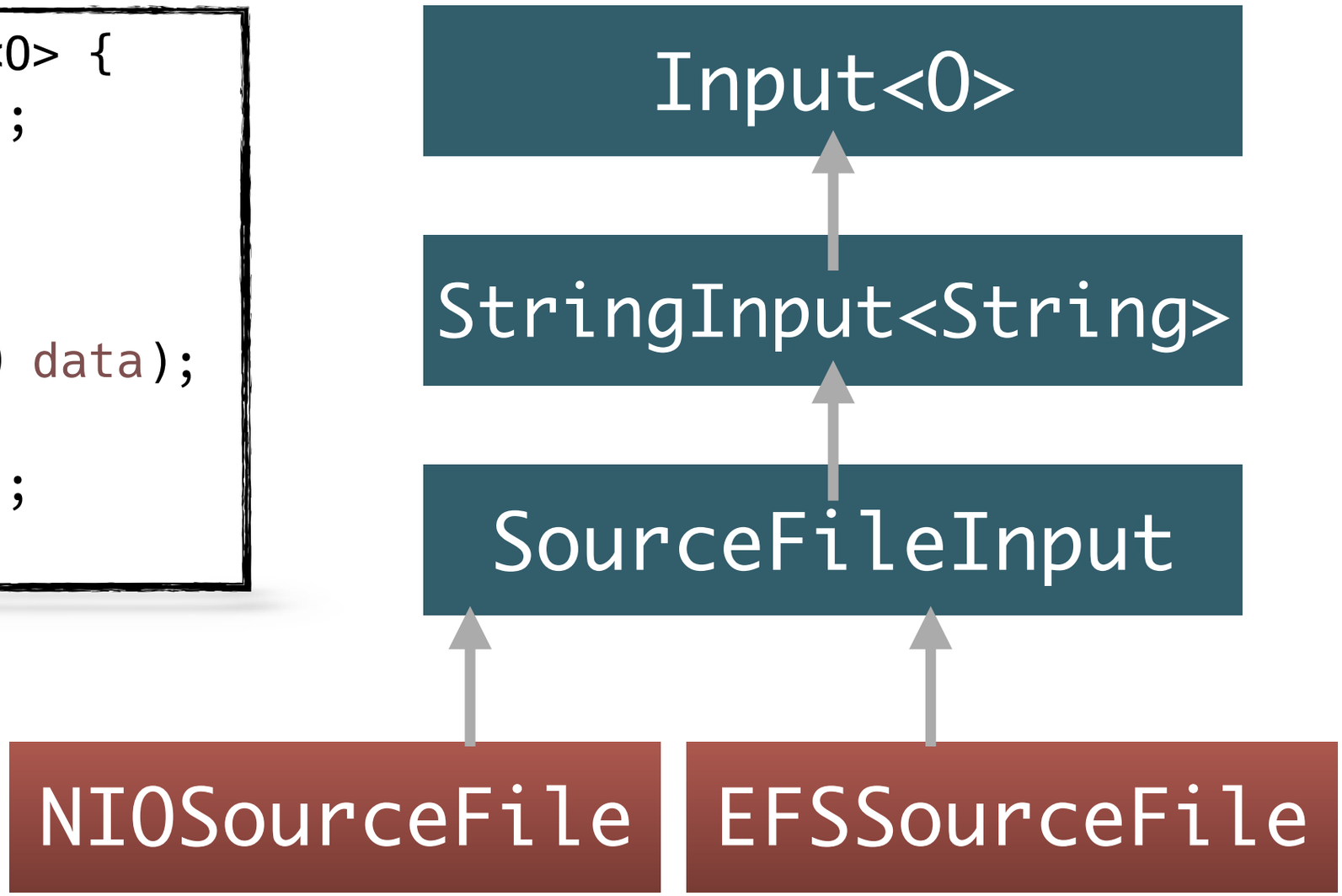


IEditorInput replacement

BestSolution

- Basic API is named `Input<O>`

```
public interface Input<O> {
  public void dispose();

  public O getData();

  public void setData(O data);

  public void persist();
}
```



Text Parsing/Highlighting

Text Parsing/Highlighting

- Stay with Eclipse Text!

Text Parsing/Highlighting

- Stay with Eclipse Text!
  - many parts are UI-Toolkit agnostic

Text Parsing/Highlighting

- Stay with Eclipse Text!
  - many parts are UI-Toolkit agnostic
  - Eclipse text does not require OSGi / 3.x APIs

Text Parsing/Highlighting

- Stay with Eclipse Text!
  - many parts are UI-Toolkit agnostic
  - Eclipse text does not require OSGi / 3.x APIs
- It is fast

Text Parsing/Highlighting

- Stay with Eclipse Text!
  - many parts are UI-Toolkit agnostic
  - Eclipse text does not require OSGi / 3.x APIs
  - It is fast
  - It does its job

Text Parsing/Highlighting

- Stay with Eclipse Text!
  - many parts are UI-Toolkit agnostic
  - Eclipse text does not require OSGi / 3.x APIs
  - It is fast
  - It does its job
  - It is insanely fast

Eclipse Text Performance

- In JavaFX world there are 2 opensource controls for StyledTextArea (from e(fx)clipse), RichTextArea on github

- Both Frameworks have a pluggable Parser Infrastructure and provide default/sample parsers

Eclipse Text Performance

- In JavaFX world there are 2 opensource controls for StyledTextArea (from e(fx)clipse), RichTextArea on github

- Both Frameworks have a pluggable Parser Infrastructure and provide default/sample parsers

	StyleText (Eclipse Text)	RichText (regex)

Eclipse Text Performance

- In JavaFX world there are 2 opensource controls for StyledTextArea (from e(fx)clipse), RichTextArea on github

- Both Frameworks have a pluggable Parser Infrastructure and provide default/sample parsers

	StyleText (Eclipse Text)	RichText (regex)
init - 10 000 Loc	327ms	400ms

Eclipse Text Performance

- In JavaFX world there are 2 opensource controls for StyledTextArea (from e(fx)clipse), RichTextArea on github

- Both Frameworks have a pluggable Parser Infrastructure and provide default/sample parsers

	StyleText (Eclipse Text)	RichText (regex)
init - 10 000 Loc	327ms	400ms
init - 150 000 Loc	1100ms	3300ms

Eclipse Text Performance

- In JavaFX world there are 2 opensource controls for StyledTextArea (from e(fx)clipse), RichTextArea on github

- Both Frameworks have a pluggable Parser Infrastructure and provide default/sample parsers

	StyleText (Eclipse Text)	RichText (regex)
init - 10 000 Loc	327ms	400ms
init - 150 000 Loc	1100ms	3300ms
change - 10 000 Loc	30ms	110ms (*)

Eclipse Text Performance

BestSolution

- In JavaFX world there are 2 opensource controls for StyledTextArea (from e(fx)clipse), RichTextArea on github

- Both Frameworks have a pluggable Parser Infrastructure and provide default/sample parsers

	StyleText (Eclipse Text)	RichText (regex)
init - 10 000 Loc	327ms	400ms
init - 150 000 Loc	1100ms	3300ms
change - 10 000 Loc	30ms	110ms (*)
change - 150 000 Loc	50ms	1800ms (*)

Eclipse Text Performance

BestSolution

- In JavaFX world there are 2 opensource controls for StyledTextArea (from e(fx)clipse), RichTextArea on github

- Both Frameworks have a pluggable Parser Infrastructure and provide default/sample parsers

	StyleText (Eclipse Text)	RichText (regex)
init - 10 000 Loc	327ms	400ms
init - 150 000 Loc	1100ms	3300ms
change - 10 000 Loc	30ms	110ms (*)
change - 150 000 Loc	50ms	1800ms (*)

* Potential Bug: numbers might be devided by 2

Eclipse Text in a nutshell

- Two step process

Eclipse Text in a nutshell

- Two step process
  - Step 1: Partitioning done by the `org.eclipse.jface.text.IDocumentPartitioner`

Eclipse Text in a nutshell

- Two step process
  - Step 1: Partitioning done by the `org.eclipse.jface.text.IDocumentPartitioner`
  - Step 2: Tokenizing done by the `org.eclipse.jface.text.presentation.IPresentationReconciler`

Partitioning


```
/**
 * This is a sample class
 */
class Sample {
   //Some information

   public void test() {
      /*
       * Some more information
       */
      var s =
        "Hello World"

        ;
      print(s);

   }
}
```

Partitioning

BestSolution

```dart
/**
 * This is a sample class
 */
class Sample {
   //Some information

   public void test() {
      /*
       * Some more information
       */
      var s =
        "Hello World"

        ;
      print(s);

   }
}
```

__dart_dartdoc

Partitioning


```
/**
 * This is a sample class
 */
class Sample {
   //Some information

   public void test() {
      /*
       * Some more information
       */
      var s =
        "Hello World"

        ;
      print(s);

   }
}
```

__dart_dartdoc

__dart_sl_comment

Partitioning

BestSolution

```
/**
 * This is a sample class
 */
class Sample {
   //Some information

   public void test() {
      /*
       * Some more information
       */
      var s =
        "Hello World"

        ;
      print(s);

   }
}
```

__dart_dartdoc

__dart_sl_comment

__dart_ml_comment

Partitioning

BestSolution

```
/**
 * This is a sample class
 */
class Sample {
   //Some information

   public void test() {
      /*
       * Some more information
       */
      var s =
        "Hello World"

        ;
      print(s);

   }
}
```

__dart_dartdoc

__dart_sl_comment

__dart_ml_comment

__dart_string

Partitioning

```
/**
 * This is a sample class
 */
class Sample {
   //Some information

   public void test() {
      /*
       * Some more information
       */
      var s =
        "Hello World"

        ;
      print(s);

   }
}
```

__dart_dartdoc

__dart_sl_comment

__dart_ml_comment

__dart_string

__dftl_partitioning

Partitioning Rules

Partitioning Rules

BestSolution

	Start	End	MultiLine

Partitioning Rules

BestSolution

	Start	End	MultiLine
dartdoc	/**	*/	✔

Partitioning Rules

BestSolution

	Start	End	MultiLine
dartdoc	/**	*/	✔
comment single	//		✖

Partitioning Rules

	Start	End	MultiLine
dartdoc	/**	*/	✔
comment single	//		✖
comment multi	/*	*/	✔

Partitioning Rules

	Start	End	MultiLine
dartdoc	/**	*/	✔
comment single	//		✖
comment multi	/*	*/	✔
string	"	"	✖

Partitioning Rules

	Start	End	MultiLine
dartdoc	/**	*/	✔
comment single	//		✖
comment multi	/*	*/	✔
string	"	"	✖
character	'	'	✖

Rules in Eclipse Text

- Base interface `org.eclipse.jface.text.rules.IRule`
- 2 Basic implementations
  - SingleLineRule
  - MultiLineRule

Rules in Eclipse Text

- Base interface `org.eclipse.jface.text.rules.IRule`
- 2 Basic implementations
  - SingleLineRule
  - MultiLineRule

```
new SingleLineRule("//", null, new Token("__dart_sl_comment"));
new MultiLineRule("/**", "*/", new Token("__dart_dartdoc");
```

From Rules to Partitioner

- Eclipse Text provides a default implementation for the `IDocumentParitioner` named `FastPartitioner`

- `FastPartitioner` delegates the real partitioning to an `IPartitionTokenScanner`

- Eclipse Text provides a default implementation for `IPartitionScanner` named `RuleBasedPartitionScanner`

- `RuleBasedPartitionScanner` use `IRule` definitions to detect partitions

From Rules to Partitioner

From Rules to Partitioner


```
public class DartPartitionScanner extends RuleBasedPartitionScanner {
  IPredicateRule[] pr = new org.eclipse.jface.text.rules.IPredicateRule[6];
  pr[0] = new SingleLineRule("//", null, new Token("__dart_sl_comment");
  // …
}
```

From Rules to Partitioner

BestSolution

```
public class DartPartitionScanner extends RuleBasedPartitionScanner {
  IPredicateRule[] pr = new org.eclipse.jface.text.rules.IPredicateRule[6];
  pr[0] = new SingleLineRule("//", null, new Token("__dart_sl_comment");
  // …
}

public class DartPartitioner extends FastPartitioner {
  public DartPartitioner() {
    super(new DartPartitionScanner(), new String[] {
      "__dart_singlelinedoc_comment"
     ,"__dart_dartdoc"
     ,"__dart_sl_comment"
     ,"__dart_multiline_comment"
     ,"__dart_string"
    });
  }
```

From Rules to Partitioner

BestSolution

```
public class DartPartitionScanner extends RuleBasedPartitionScanner {
  IPredicateRule[] pr = new org.eclipse.jface.text.rules.IPredicateRule[6];
  pr[0] = new SingleLineRule("//", null, new Token("__dart_sl_comment");
  // …
}

public class DartPartitioner extends FastPartitioner {
  public DartPartitioner() {
    super(new DartPartitionScanner(), new String[] {
      "__dart_singlelinedoc_comment"
     ,"__dart_dartdoc"
     ,"__dart_sl_comment"
     ,"__dart_multiline_comment"
     ,"__dart_string"
    });
  }
}
```

Express Partitions in a DSL

- Instead of writing a bunch of Java configuration code we could define a DSL and generate the code

Express Partitions in a DSL

BestSolution

- Instead of writing a bunch of Java configuration code we could define a DSL and generate the code

```
package langs

dart {
    partitioning {
        partition __dftl_partition_content_type
        partition __dart_singlelinedoc_comment
        partition __dart_dartdoc
        partition __dart_sl_comment
        partition __dart_multiline_comment
        partition __dart_string
        rule {
            single_line __dart_string                "'" => "'" escaped by "\\"
            single_line __dart_string                '"' => '"' escaped by "\\"
            single_line __dart_singlelinedoc_comment '///'
            single_line __dart_sl_comment            '//'
            multi_line  __dart_dartdoc               '/**' => '*/'
            multi_line  __dart_multiline_comment     '/*'  => '*/'
        }
    }
…
}
```

Tokenizing a Partition

BestSolution

```
/**
 * This is a sample class
 */
class Sample {

   public void test(String s) {
      print(s);
   }

}
```

Tokenizing a Partition


```
/**
 * This is a sample class
 */
class Sample {

   public void test(String s) {
      print(s);
   }

}
```

tk(dart_doc,0,25)

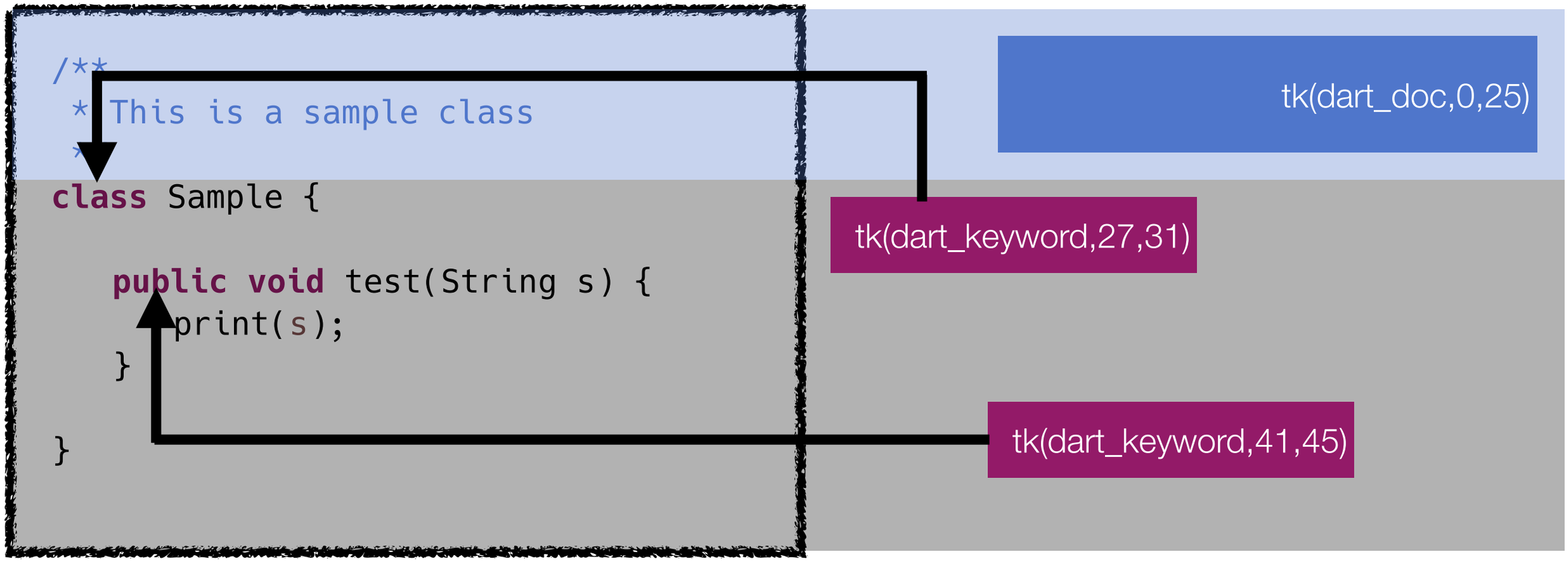
Tokenizing a Partition

BestSolution

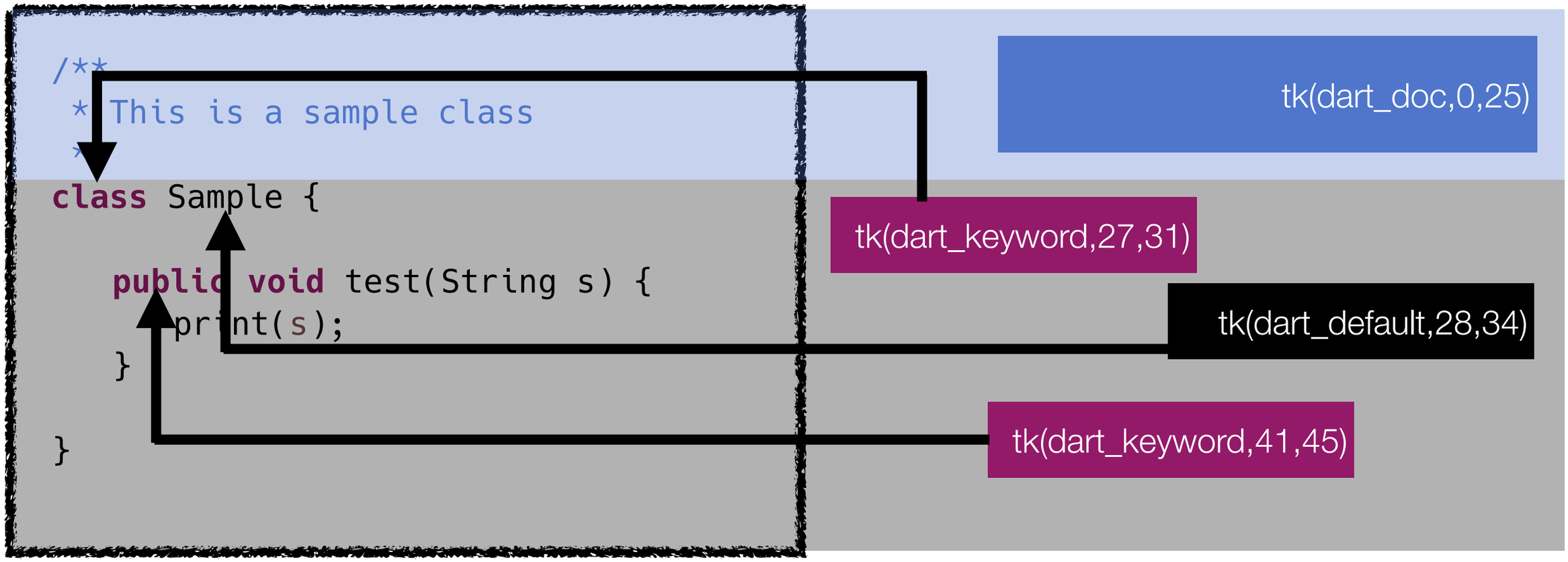
```
/**
 * This is a sample class
 */
class Sample {

   public void test(String s) {
      print(s);
   }

}
```

tk(dart_doc,0,25)

Tokenizing a Partition



Tokenizing a Partition



Tokenizing Rules

- Uses same `IRules` APIs as for partitioning
- Prebuilt rules useable for Tokenizing
  - `CombinedWordRule`: Used for keywords
  - `CharacterRule`: Used for single chars eg braces, operators
  - `SingleLineRule`, `MultiLineRule`, `PatternRule`, …

From Rules to Reconciler

- Eclipse Text provides a default implementation for `IPresentationReconciler` named `PresentationReconciler`
- `PresentationReconciler` requires an `IPresentationDamager` and `IPresentationRepairer` / Partition
- Eclipse Text provides a default implementation for `IPresentationDamager` & `IPresentationRepairer` named `DefaultDamagerRepairer`
- `DefaultDamagerRepairer` uses an `ITokenScanner`
- Eclipse Text provide a default implementation `ITokenScanner` named `RuleBasedScanner` who uses `IRule`

From Rules to Reconciler

From Rules to Reconciler

BestSolution

```
public class DartDefPartitionScanner extends RuleBasedScanner {
  public DartDefPartitionScanner() {
    Token dart_defaultToken = new Token(new TextAttribute("dart.dart_default"));
    setDefaultReturnToken(dart_defaultToken);
    Token dart_keywordToken = new Token(new TextAttribute("dart.dart_keyword"));

    JavaLikeWordDetector wordDetector= new JavaLikeWordDetector();
    CombinedWordRule combinedWordRule= new CombinedWordRule(wordDetector, dart_defaultToken);
    CombinedWordRule.WordMatcher dart_keywordWordRule = new CombinedWordRule.WordMatcher();
    dart_keywordWordRule.addWord("break", dart_keywordToken);
    // …
    combinedWordRule.addWordMatcher(dart_keywordWordRule);
  }
}
```

From Rules to Reconciler

BestSolution

```
public class DartDefPartitionScanner extends RuleBasedScanner {
  public DartDefPartitionScanner( ) {
    Token dart_defaultToken = new Token(new TextAttribute("dart.dart_default"));
    setDefaultReturnToken(dart_defaultToken);
    Token dart_keywordToken = new Token(new TextAttribute("dart.dart_keyword"));

    JavaLikeWordDetector wordDetector= new JavaLikeWordDetector( );
    CombinedWordRule combinedWordRule= new CombinedWordRule(wordDetector, dart_defaultToken);
    CombinedWordRule.WordMatcher dart_keywordWordRule = new CombinedWordRule.WordMatcher( );
    dart_keywordWordRule.addWord("break", dart_keywordToken);
    // …
    combinedWordRule.addWordMatcher(dart_keywordWordRule);
  }
}

public class DartPresentationReconciler extends PresentationReconciler {
  public DartPresentationReconciler() {
    DefaultDamagerRepairer defDamageRepairer = new DefaultDamagerRepairer(new DartDefPartitionScanner());
    setDamager(defDamageRepairer, "__dftl_partition_content_type");
    setRepairer(defDamageRepairer, "__dftl_partition_content_type");

    DefaultDamagerRepairer docDamageRepairer = new DefaultDamagerRepairer(new DartDocPartitionScanner());
    setDamager(docDamageRepairer, "__dart_dartdoc");
    setRepairer(docDamageRepairer, "__dart_dartdoc");
    // …
  }
}
```

Express Tokens in the DSL

Express Tokens in the DSL


```
package langs

dart {
  partitioning {
    partition __dftl_partition_content_type
    partition __dart_dartdoc
    // …
  }
```

Express Tokens in the DSL


```
package langs

dart {
  partitioning {
    partition __dftl_partition_content_type
    partition __dart_dartdoc
    // …
  }
  lexical_highlighting {
    rule __dftl_partition_content_type {
      default dart_default
      dart_keyword {
        keywords [ "break", "case", /* … */ ]
      }
    }
    rule __dart_dartdoc {
      default dart_doc
      dart_doc_reference {
        single_line "[" => "]"
      }
    }
  }
}
```

Visual representation

Visual representation


```
public class DartDefPartitionScanner extends RuleBasedScanner {
  public DartDefPartitionScanner() {
    Token dart_defaultToken = new Token(new TextAttribute("dart.dart_default"));
    setDefaultReturnToken(dart_defaultToken);
    Token dart_keywordToken = new Token(new TextAttribute("dart.dart_keyword"));

    // …
```

Visual representation

BestSolution

```
public class DartDefPartitionScanner extends RuleBasedScanner {
  public DartDefPartitionScanner() {
    Token dart_defaultToken = new Token(new TextAttribute("dart.dart_default"));
    setDefaultReturnToken(dart_defaultToken);
    Token dart_keywordToken = new Token(new TextAttribute("dart.dart_keyword"));

    // …
```

```
// … somewhere in StyledTextArea
StyledTextNode s = new StyledTextNode("public");
s.getStyleClass().setAll("dart","dart_keyword");
// … somewhere in StyledTextArea
```

Visual representation


```
public class DartDefPartitionScanner extends RuleBasedScanner {
  public DartDefPartitionScanner() {
    Token dart_defaultToken = new Token(new TextAttribute("dart.dart_default"));
    setDefaultReturnToken(dart_defaultToken);
    Token dart_keywordToken = new Token(new TextAttribute("dart.dart_keyword"));

    // …
```

```
// … somewhere in StyledTextArea
StyledTextNode s = new StyledTextNode("public");
s.getStyleClass().setAll("dart","dart_keyword");
// … somewhere in StyledTextArea
```

```
.styled-text-area .dart.dart_keyword {
    -styled-text-color: rgb(127, 0, 85);
    -fx-font-weight: bold;
}
```

Visual representation


```
public class DartDefPartitionScanner extends RuleBasedScanner {
  public DartDefPartitionScanner() {
    Token dart_defaultToken = new Token(new TextAttribute("dart.dart_default"));
    setDefaultReturnToken(dart_defaultToken);
    Token dart_keywordToken = new Token(new TextAttribute("dart.dart_keyword"));

    // …
```

```
// … somewhere in StyledTextArea
StyledTextNode s = new StyledTextNode("public");
s.getStyleClass().setAll("dart","dart_keyword");
// … somewhere in StyledTextArea
```

```
.styled-text-area .dart.dart_keyword {
    -styled-text-color: -source-editor-keyword;
    -fx-font-weight: bold;
}
```

Live Demo (Lexicalhighlighting)

Eclipse 4 Architecture

BestSolution

Eclipse 4 Architecture

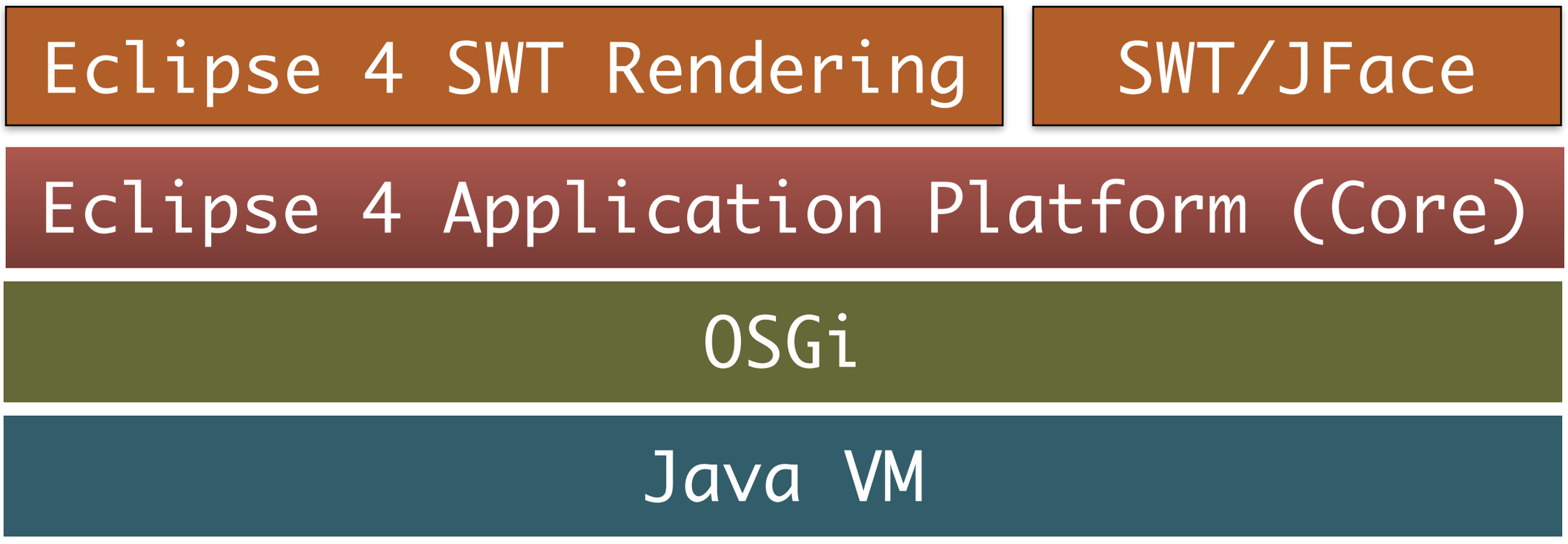
Java VM

Eclipse 4 Architecture

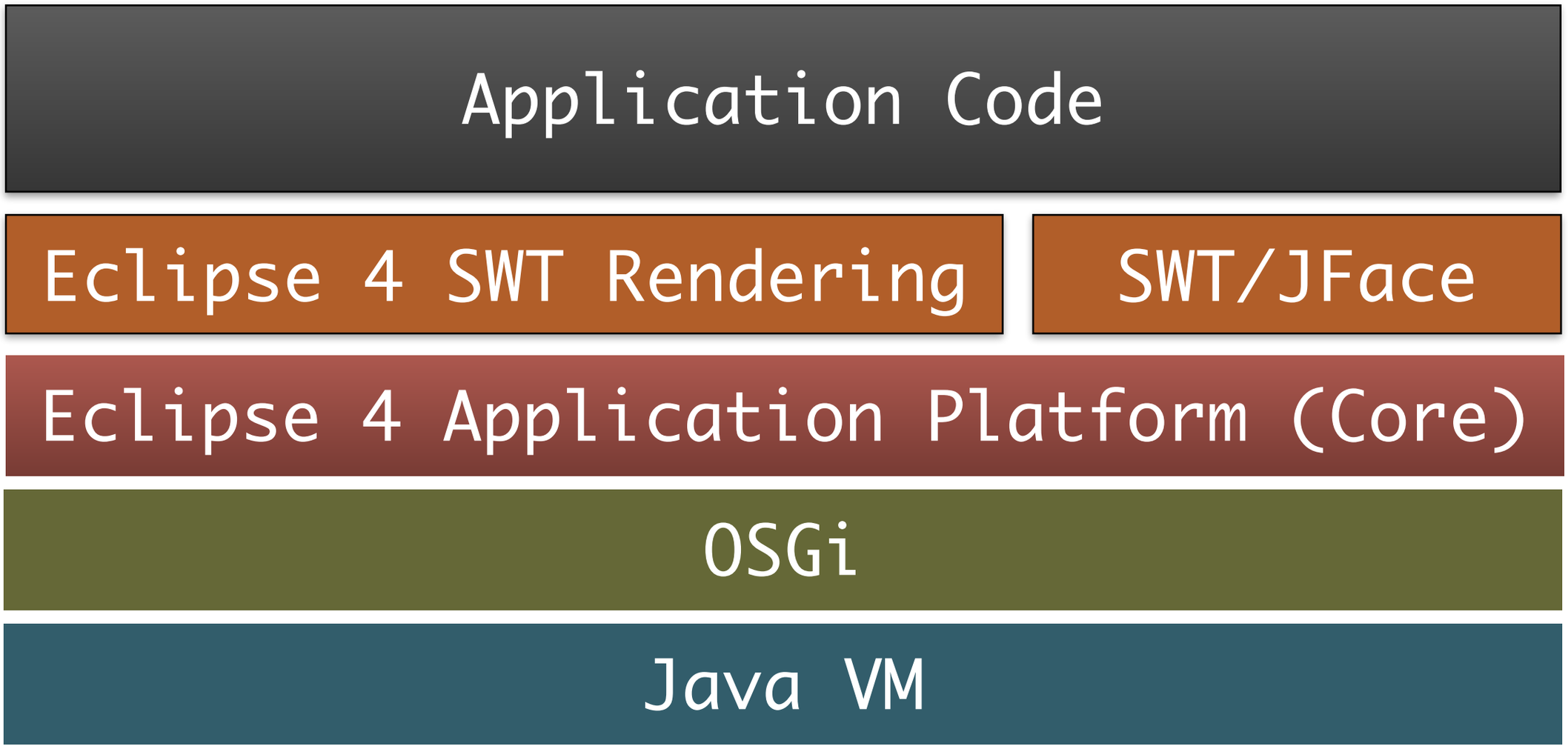
BestSolution

Eclipse 4 Architecture

Eclipse 4 Architecture

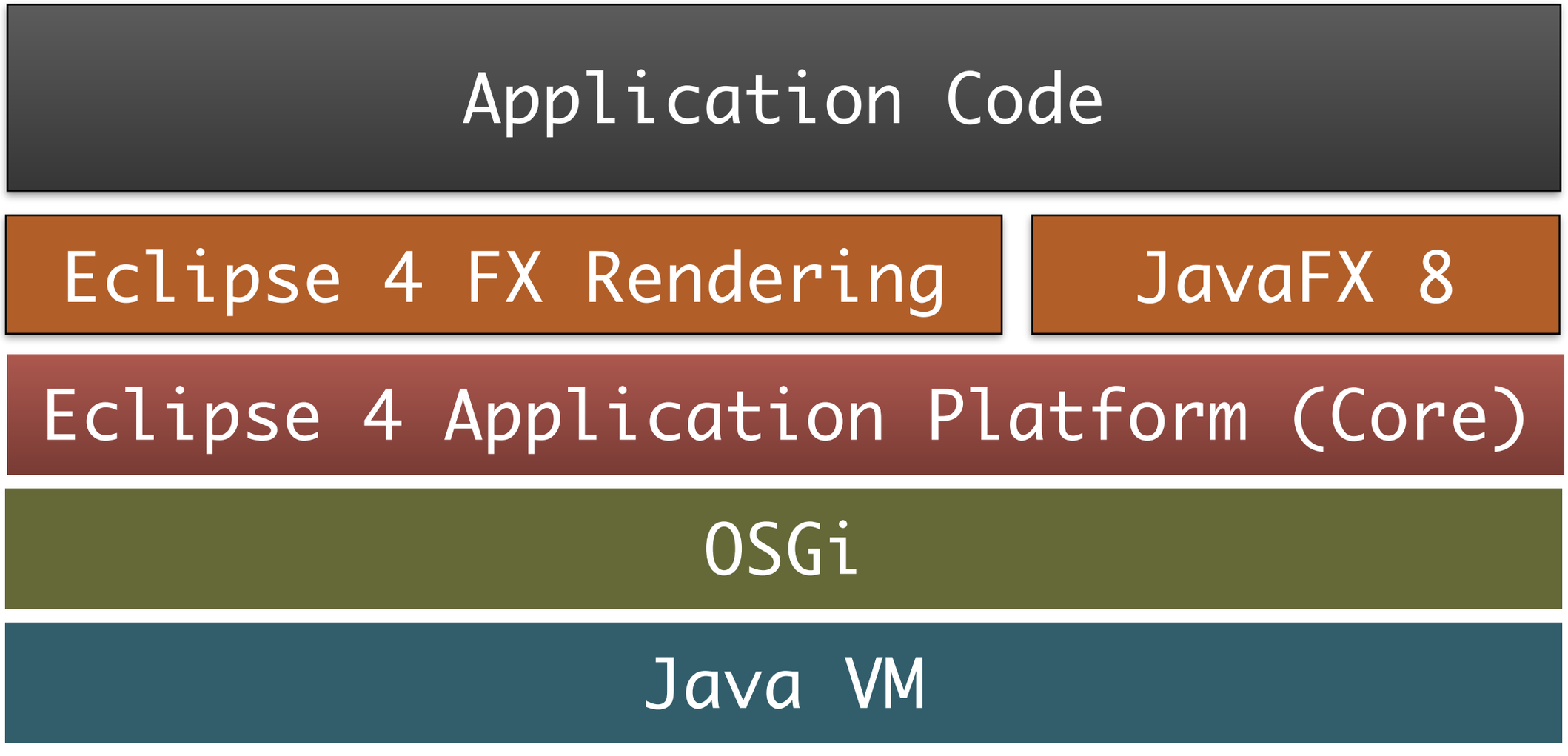


Eclipse 4 Architecture

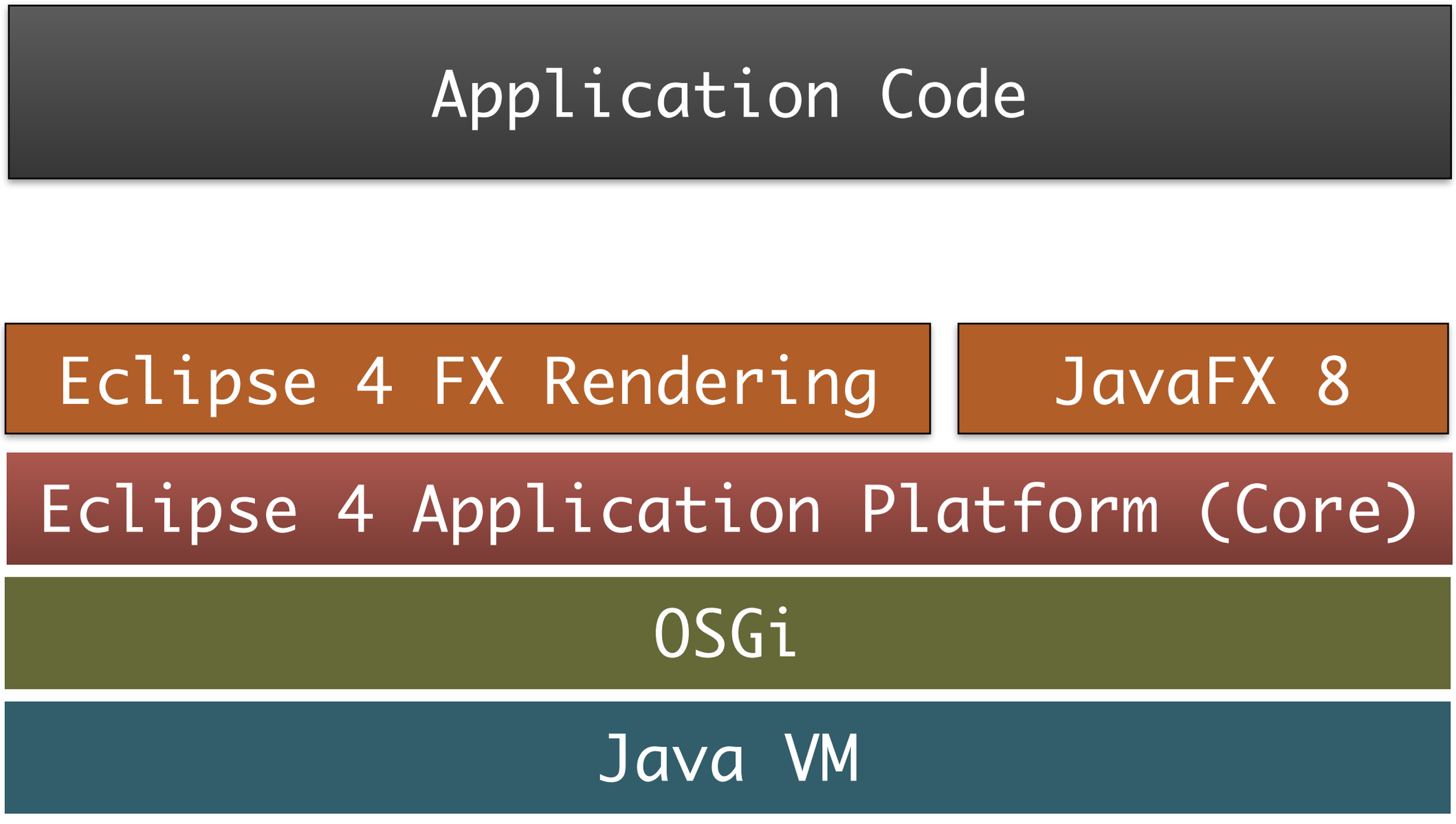


Eclipse 4 Architecture

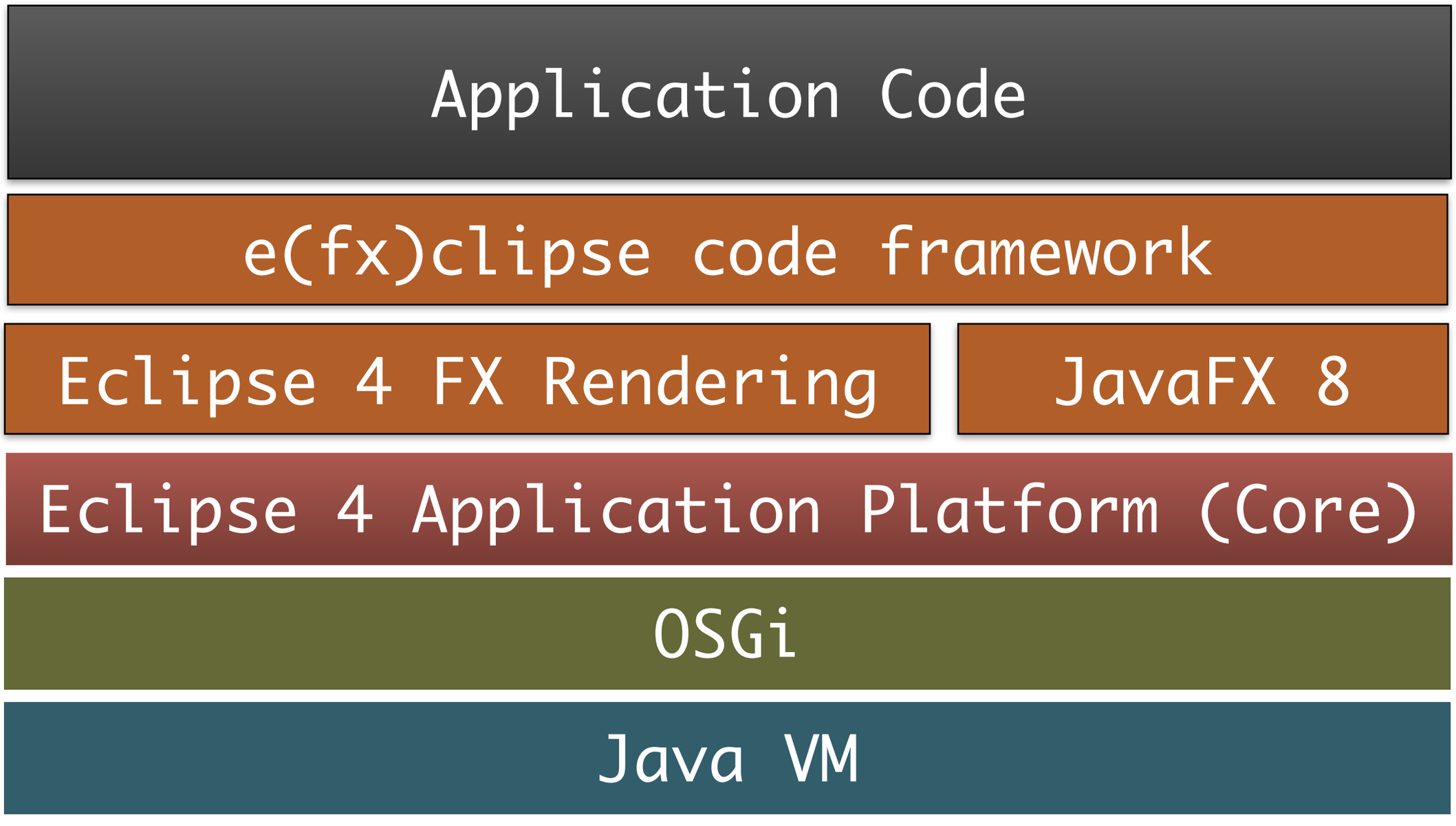
Eclipse 4 Architecture



Eclipse 4 Architecture



Eclipse 4 Architecture



Integration in Eclipse 4

- The task: We need to create the correct `IDocumentPartitioner` and `IPresentationReconciler` for a given Source-File

- e(fx)clipse defines a `TypeProviderService` Service-API who is consulted to find a type for a given `Input` (=SourceFile)

- ALL `TypeProviderService`-Instances are registered in the OSGi-Service-Registry and consulted by the framework when it requires an instance of `IDocumentPartitioner` or `IPresentationReconciler`

Integration into e4

- The `TypeProviderService` APIs

Integration into e4

BestSolution

- The `TypeProviderService` APIs

```
/**
 * A service who provides a type based upon a selector value
 *
 * @param <S>
 *            the selector value type
 * @param <T>
 *            the type
 * @since 2.1
 */
public interface TypeProviderService<S, T> extends Predicate<S> {
    @Override
    boolean test(S t);

    public Class<? extends T> getType(S s);
}

interface InputDependentTypeProviderService<T> extends TypeProviderService<Input<?>, T> {
}

interface DocumentPartitionerTypeProvider extends
    InputDependentTypeProviderService<IDocumentPartitioner> {
}
```

Integration in e4

Integration in e4

BestSolution

```
package org.eclipse.fx.code.editor.fx;

/**
 * Component setting up a JavaFX text editor
 */
@SuppressWarnings("restriction")
public class TextEditor {

  @Inject
  public void setPartitioner(IDocumentPartitioner partitioner) {

  }
}
```

Integration in e4

```java
package org.eclipse.fx.code.editor.fx;

/**
 * Component setting up a JavaFX text editor
 */
@SuppressWarnings("restriction")
public class TextEditor {

  @Inject
  public void setPartitioner(IDocumentPartitioner partitioner) {

  }
}
```

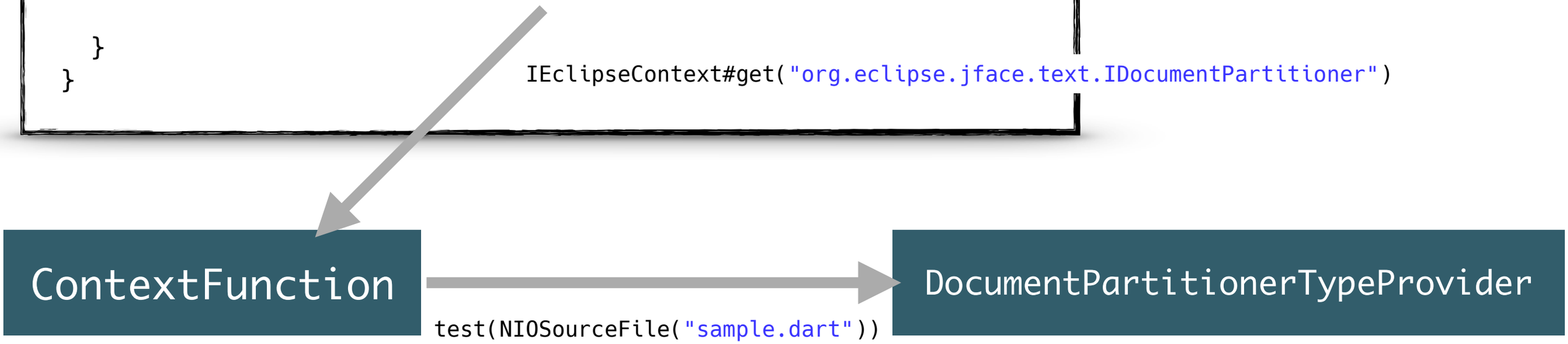
IEclipseContext#get("org.eclipse.jface.text.IDocumentPartitioner")

ContextFunction

Integration in e4

```
package org.eclipse.fx.code.editor.fx;

/**
 * Component setting up a JavaFX text editor
 */
@SuppressWarnings("restriction")
public class TextEditor {

  @Inject
  public void setPartitioner(IDocumentPartitioner partitioner) {

  }
}
```

IEclipseContext#get("org.eclipse.jface.text.IDocumentPartitioner")



Integration in e4

Integration in e4

BestSolution

```
@Component
public class DartDocumentPartitionerTypeProvider implements DocumentPartitionerTypeProvider {

    public Class<? extends IDocumentPartitioner> getType(Input<?> s) {
        return DartPartitioner.class;
    }

    public boolean test(Input<?> t) {
        return (t instanceof URIProvider) && ((URIProvider)t).getURI().lastSegment().endsWith(".dart");
    }
}
```

Integration in e4


```
@Component
public class DartDocumentPartitionerTypeProvider implements DocumentPartitionerTypeProvider {

    public Class<? extends IDocumentPartitioner> getType(Input<?> s) {
        return DartPartitioner.class;
    }

    public boolean test(Input<?> t) {
        return (t instanceof URIProvider) && ((URIProvider)t).getURI().lastSegment().endsWith(".dart");
    }
}

@Component
public class DartPresentationReconcilerTypeProvider implements PresentationReconcilerTypeProvider {

    public Class<? extends PresentationReconciler> getType(Input<?> s) {
        return DartPresentationReconciler.class;
    }

    public boolean test(Input<?> t) {
        return (t instanceof URIProvider) && ((URIProvider)t).getURI().lastSegment().endsWith(".dart");
    }
}
```

Autocomplete

BestSolution

- There’s a trend of having headless applications providing things like autocomplete, error reporting, …
  - Dart: DartAnalysis Server
  - Typescript: LanguageService in tsserver
  - Java:
    - Eclipse Flux, Eclipse Che (Cloudbased)
    - CodeSurf: internal research project

Autocomplete for Dart

BestSolution

- Communication with DartAnalysisServer is via JSON on STDIN/STDOUT

- 2 Interaction Types:
  - Requests with a direct response (client driven)
  - Notification informing about things (server driven)

Interact with Dart Server

BestSolution

1. Step

Inform DartServer about the source dir

```dart
class Rectangle {
  num left;
  num top;
  num width;
  num height;

  num get right             => left + width;
      set right(num value)  => left = value - width;
  num get bottom            => top + height;
      set bottom(num value) => top = value - height;
}

// This is where the app starts executing.
main() {
  var r = new Rectangle();
  r.
}
```

```
{
  "id" : "default_1",
  "method" : "analysis.setAnalysisRoots" ,
  "params" :  {
    "included":["/Users/tomschindl/dart-samples/"],
    "excluded":[]
  }
}
```

Interact with Dart Server

BestSolution

2. Step

Request autocompletion

```dart
class Rectangle {
  num left;
  num top;
  num width;
  num height;

  num get right             => left + width;
      set right(num value)  => left = value - width;
  num get bottom            => top + height;
      set bottom(num value) => top = value - height;
}

// This is where the app starts executing.
main() {
  var r = new Rectangle();
  r.
}
```

```
{
  "id" : "default_2",
  "method" : "completion.getSuggestions" ,
  "params" :  {
    "file":"/Users/tomschindl/dart-samples/test.dart",
    "offset":367
  }
}
```

Interact with Dart Server

BestSolution


```
{
  "event":"completion.results",
   "params":{
     "id":"0",
     "replacementOffset":367,
     "replacementLength":0,
     "results": [
       {
         "kind":"INVOCATION",
         "relevance":1000,
         "completion":"left",
         "selectionOffset":4,
         "selectionLength":0,
         "isDeprecated":false,
         "isPotential":false,
         "declaringType":"Rectangle",
         "element":{
           "kind":"FIELD",
           "name":"left",
           "location":{
             "file":"/Users/tomschindl/dart-samples/test.dart",
             "offset":24,
             "length":4,
             "startLine":2,
             "startColumn":7
           },
           "flags":0,
           "returnType":"num"
         },"returnType":"num"
       },
       // Many more …

     "isLast":true
   }
}
```


3. Step

Server asynchronously delivers completion results

Interact with Dart Server

- Interaction with Java Code

Interact with Dart Server

- Interaction with Java Code

```
DartServerFactory serverFactory = Util.lookupService(DartServerFactory.class);
DartServer server = serverFactory.getServer("server");
```

Interact with Dart Server

- Interaction with Java Code

```
DartServerFactory serverFactory = Util.lookupService(DartServerFactory.class);
DartServer server = serverFactory.getServer("server");

ServiceAnalysis analysisService = server.getService(ServiceAnalysis.class);
ServiceCompletion completionService = server.getService(ServiceCompletion.class);
```

Interact with Dart Server

- Interaction with Java Code

```
DartServerFactory serverFactory = Util.lookupService(DartServerFactory.class);
DartServer server = serverFactory.getServer("server");

ServiceAnalysis analysisService = server.getService(ServiceAnalysis.class);
ServiceCompletion completionService = server.getService(ServiceCompletion.class);

analysisService.setAnalysisRoots(new String[] {"/Users/tomschindl/dart-samples/"}, new String[0], null);
```

Interact with Dart Server

- Interaction with Java Code

```
DartServerFactory serverFactory = Util.lookupService(DartServerFactory.class);
DartServer server = serverFactory.getServer("server");

ServiceAnalysis analysisService = server.getService(ServiceAnalysis.class);
ServiceCompletion completionService = server.getService(ServiceCompletion.class);

analysisService.setAnalysisRoots(new String[] {"/Users/tomschindl/dart-samples/"}, new String[0], null);

Registration proposalRegistration = completionService.results(this::handleHandleResults);
completionService.getSuggestions("/Users/tomschindl/dart-samples/test.dart", 367);

private static void handleHandleResults(CompletionResultsNotification notification) {
  Stream.of(notification.getResults()).forEach( c -> System.err.println(c.getCompletion()));
}
```

Code Editor Autocomplete

Code Editor Autocomplete

- Autocomplete is implemented by a service of type `org.eclipse.fx.code.editor.fx.services.ProposalComputer`

Code Editor Autocomplete

- Autocomplete is implemented by a service of type `org.eclipse.fx.code.editor.fx.services.ProposalComputer`
- and registered in the OSGi-Service registry through `org.eclipse.fx.code.editor.fx.services.ProposalComputerTypeProvider`

Code Editor Autocomplete

BestSolution

- Autocomplete is implemented by a service of type `org.eclipse.fx.code.editor.fx.services.ProposalComputer`
- and registered in the OSGi-Service registry through `org.eclipse.fx.code.editor.fx.services.ProposalComputerTypeProvider`

```
public class DartProposalComputer implements ProposalComputer {
  //…
  @Override
  public Future<List<ICompletionProposal>> compute(ProposalContext context) {
    URIProvider p = (URIProvider) context.input;
    Path file = Paths.get(java.net.URI.create(p.getURI().toString())).toAbsolutePath();

    CompletionGetSuggestionsResult result = completionService.getSuggestions(file.toString(),
      context.location);
    requestId = result.getId();

    future = new CompletableFuture<>();

    return future;
  }
}
```

Live Demo (Autocomplete)

- Syntax Highlighting
- Autocomplete
- Error Reporting

Error Markers

- DartServer provides the possibility to
  - subscribe to error notifications
  - fetch the current list of errors

Error Markers

- DartServer provides the possibility to
  - subscribe to error notifications
  - fetch the current list of errors

```
ServiceAnalysis service = server.getService(ServiceAnalysis.class);
subscription = service.errors(this::accept);

CompletableFuture.supplyAsync(
    () -> service.getErrors(file.toString())).thenAccept(this::accept);
```

Code Editor Markers

- Code editor consults to services for error support:
  - `org.eclipse.jface.text.source.IAnnotationModel`: Collects all errors and stores them for later useage
  - `org.eclipse.jface.text.source.AnnotationPresenter`: Presents the errors in the TextEditor (eg as markers in the line ruler)
  - services are contributed through `org.eclipse.fx.code.editor.services.AnnotationModelTypeProvider` and `org.eclipse.fx.code.editor.fx.services.AnnotationPresenterTypeProvider`

Live Demo (Error Marker)

Language Support

- Syntaxhighlighting:
  - Opensource: asciidoc, ceylon, dart, go, groovy, java, js, kotlin, lua, php, python, rust, swift, xml
  - Internal research: Typescript
- Autocomplete:
  - Opensource: Dart
  - Internal research: Java, JavaScript, Typescript
- Outline:
  - Opensource: Dart
  - Internal research: JavaScript, Java, Typescript