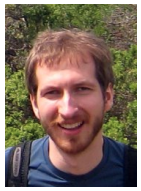
[ICSE12]

Active Code Completion

Cyrus Omar
Computer Science

YoungSeok Yoon
Software Engineering



Thomas D. LaToza
Software Engineering

Brad A. Myers
Human-Computer Interaction

School of Computer Science
Carnegie Mellon University

Code Completion

```
import java.util.regex.Pattern;

public class Matcher {
    public static boolean isTemperature(String s) {
        Pattern p = |
    }
}
```

Pattern – java.util.regex
p : Pattern
s : String
isTemperature(String s) : boolean – Matc
Matcher – edu.cmu.cs.comar

Press '^Space' to show Template Proposals

A compiled representation of a regular expression.

A regular expression, specified as a string, must first be compiled into an instance of this class. The resulting pattern can then be used to create a `Matcher` object that can match arbitrary character sequences against the regular expression. All of the state involved in performing a match resides in the matcher, so many matchers can share the same pattern.

A typical invocation sequence is thus

Press 'Tab' from proposal table or click for focus

Code Completion is Commonly Used

Table 3

Top 10 commands executed across all 41 developers

Command	Identifier	Use (%)
Delete	org.eclipse.ui.edit.delete	14.3
Save	org.eclipse.ui.file.save	11.3
Next word	org.eclipse.ui.edit.text.goto.wordNext	7.3
Paste	org.eclipse.ui.edit.paste	6.8
Content assist	org.eclipse.ui.edit.text.contentAssist.proposals	6.7
Previous word	org.eclipse.ui.edit.text.goto.wordPrevious	5.9
Copy	org.eclipse.ui.edit.copy	4.6
Select previous word	org.eclipse.ui.edit.text.select.wordPrevious	3.4
Step (debug)	org.eclipse.debug.ui.debugview.toolbar.stepOver	3.2

(Murphy et al, 2006)

Code Completion is Useful

- Helps developers explore relevant APIs
  - Avoids *context switches* to external API browsers

Code Completion is Useful

- Helps developers explore relevant APIs
  - Avoids *context switches* to external API browsers
- Helps developers avoid mistakes
  - Spelling and type errors are reduced

Code Completion is Useful

- Helps developers explore relevant APIs
  - Avoids *context switches* to external API browsers
- Helps developers avoid mistakes
  - Spelling and type errors are reduced
- Reduces required keystrokes
  - Increases the amount of information conveyed per keystroke

A *code completion system* can be characterized by the set of sources it queries to predict the user's intent.

- Language & API specs
  (current editors)

A *code completion system* can be characterized by the set of sources it queries to predict the user's intent.

- Language & API specs
  (current editors)
- Usage history
  [Robbes & Lanza, ASE08], [Hou & Pletcher, ICSM11]

A *code completion system* can be characterized by the set of sources it queries to predict the user's intent.

- Language & API specs
  (current editors)
- Usage history
  [Robbes & Lanza, ASE08], [Hou & Pletcher, ICSM11]
- Example repositories
  [Bruch et al, FSE09], [Brandt et al, CHI10], [Mooty et al, VLHCC10], [Nguyen et al, ICSE12]

A *code completion system* can be characterized by the set of sources it queries to predict the user's intent.

- Language & API specs
  (current editors)
- Usage history
  [Robbes & Lanza, ASE08], [Hou & Pletcher, ICSM11]
- Example repositories
  [Bruch et al, FSE09], [Brandt et al, CHI10], [Mooty et al, VLHCC10], [Nguyen et al, ICSE12]

A *code completion system* can be characterized by the set of sources it queries to predict the user's intent.

- Language & API specs
  (current editors)
- Usage history
  [Robbes & Lanza, ASE08], [Hou & Pletcher, ICSM11]
- Example repositories
  [Bruch et al, FSE09], [Brandt et al, CHI10], [Mooty et al, VLHCC10], [Nguyen et al, ICSE12]
- **Direct user responses**

A *code completion system* can be characterized by the set of sources it queries to predict the user's intent.

- Language & API specs
  (current editors)
- Usage history
  [Robbes & Lanza, ASE08], [Hou & Pletcher, ICSM11]
- Example repositories
  [Bruch et al, FSE09], [Brandt et al, CHI10], [Mooty et al, VLHCC10], [Nguyen et al, ICSE12]
- **Direct user responses**

…to which questions?

A *code completion system* can be characterized by the set of sources it queries to predict the user's intent.

- Language & API specs
  (current editors)
- Usage history
  [Robbes & Lanza, ASE08], [Hou & Pletcher, ICSM11]
- Example repositories
  [Bruch et al, FSE09], [Brandt et al, CHI10], [Mooty et al, VLHCC10], [Nguyen et al, ICSE12]
- **Direct user responses**
  to **domain-specific queries**

A *code completion system* can be characterized by the set of sources it queries to predict the user's intent.

- Language & API specs
  (current editors)
- Usage history
  [Robbes & Lanza, ASE08], [Hou & Pletcher, ICSM11]
- Example repositories
  [Bruch et al, FSE09], [Brandt et al, CHI10], [Mooty et al, VLHCC10], [Nguyen et al, ICSE12]
- **Direct user responses** to **domain-specific queries**

Let **domain experts** decide!

A *code completion system* can be characterized by the set of sources it queries to predict the user's intent.

- Language & API specs
  (current editors)
- Usage history
  [Robbes & Lanza, ASE08], [Hou & Pletcher, ICSM11]
- Example repositories
  [Bruch et al, FSE09], [Brandt et al, CHI10], [Mooty et al, VLHCC10], [Nguyen et al, ICSE12]
- **Direct user responses** to **domain-specific queries**
  [Omar et al, ICSE12]

= **Active Code Completion**

Active Code Completion with GRAPHITE


```
import java.util.regex.Pattern;

public class Matcher {
    public static boolean isTemperature(String s) {
        Pattern p = |
    }
}
```

- Use the regular expression workbench...
- Pattern – java.util.regex
- p : Pattern
- s : String
- isTemperature(String s) : boolean – Matc
- Matcher – edu.cmu.cs.comar

Displays a workbench that allows you to enter a regular expression pattern and test it against positive and negative examples. Automatically handles escape sequences!

Press '^Space' to show Template Proposals

Press 'Tab' from proposal table or click for focus

Active Code Completion with GRAPHITE


```
import java.util.regex.Pattern;

public class Matcher {
    public static boolean isTemperature(String s) {
        Pattern p =
    }
}
```

Enter your regular expression pattern here.

☐ Ignore Case

Should match...

*(enter **positive** test cases above, pressing ENTER between each one)*

Should NOT match...

*(enter **negative** test cases above, pressing ENTER between each one)*

Pattern	Description
.	Matches any character
^regex	Must match at the beginning of the line
regex$	Must match at the end of the line
[abc]	Set definition, matches the letter a or b or c
[abc][vz]	Set definition, matches a or b or c followed by v or z
[^abc]	Negates the pattern. Matches any character except a or b or c
[a-d1-7]	Ranges, letter between a and d or digits from 1 to 7, will not match d1
X|Z	Finds X or Z
XZ	Finds X directly followed by Z
\d	Any digit, short for [0-9]
\D	A non-digit, short for [^0-9]
\s	A whitespace character, short for [\t\n\x0b\r\f]
\S	A non-whitespace character, for short

Active Code Completion with GRAPHITE


```
import java.util.regex.Pattern;

public class Matcher {
    public static boolean isTemperature(String s) {
        Pattern p =
    }
}
```

Enter your regular expression pattern here.

☐ Ignore Case

Should match...

37F

Should NOT match...

*(enter **negative** test cases above, pressing ENTER between each one)*

= matched by pattern

Pattern	Description
.	Matches any character
^regex	Must match at the beginning of the line
regex$	Must match at the end of the line
[abc]	Set definition, matches the letter a or b or c
[abc][vz]	Set definition, matches a or b or c followed by v or z
[^abc]	Negates the pattern. Matches any character except a or b or c
[a-d1-7]	Ranges, letter between a and d or digits from 1 to 7, will not match d1
X|Z	Finds X or Z
XZ	Finds X directly followed by Z
\d	Any digit, short for [0-9]
\D	A non-digit, short for [^0-9]
\s	A whitespace character, short for [\t\n\x0b\r\f]
\S	A non-whitespace character, for short

Active Code Completion with GRAPHITE


```
import java.util.regex.Pattern;

public class Matcher {
    public static boolean isTemperature(String s) {
        Pattern p =
    }
}
```

Enter your regular expression pattern here.

☐ Ignore Case

Should match...

37F
42.1 F
.8C
-10C

Should NOT match...

*(enter **negative** test cases above, pressing ENTER between each one)*

= matched by pattern

Pattern	Description
.	Matches any character
^regex	Must match at the beginning of the line
regex$	Must match at the end of the line
[abc]	Set definition, matches the letter a or b or c
[abc][vz]	Set definition, matches a or b or c followed by v or z
[^abc]	Negates the pattern. Matches any character except a or b or c
[a-d1-7]	Ranges, letter between a and d or digits from 1 to 7, will not match d1
X|Z	Finds X or Z
XZ	Finds X directly followed by Z
\d	Any digit, short for [0-9]
\D	A non-digit, short for [^0-9]
\s	A whitespace character, short for [\t\n\x0b\r\f]
\S	A non-whitespace character, for short

Active Code Completion with GRAPHITE


```
import java.util.regex.Pattern;

public class Matcher {
    public static boolean isTemperature(String s) {
        Pattern p =
    }
}
```

Enter your regular expression pattern here.

☐ Ignore Case

Should match...	Should NOT match...
37F	12:05
42.1 F	37
.8C	37Q
-10C	

= matched by pattern

Pattern	Description
.	Matches any character
^regex	Must match at the beginning of the line
regex$	Must match at the end of the line
[abc]	Set definition, matches the letter a or b or c
[abc][vz]	Set definition, matches a or b or c followed by v or z
[^abc]	Negates the pattern. Matches any character except a or b or c
[a-d1-7]	Ranges, letter between a and d or digits from 1 to 7, will not match d1
X|Z	Finds X or Z
XZ	Finds X directly followed by Z
\d	Any digit, short for [0-9]
\D	A non-digit, short for [^0-9]
\s	A whitespace character, short for [\t\n\x0b\r\f]
\S	A non-whitespace character, for short

Active Code Completion with GRAPHITE


```
import java.util.regex.Pattern;

public class Matcher {
    public static boolean isTemperature(String s) {
        Pattern p =
    }
}
```

^|$

Ignore Case

Should match...	Should NOT match...
37F	12:05
42.1 F	37
.8C	37Q
-10C	

= matched by pattern

Pattern	Description
.	Matches any character
^regex	Must match at the beginning of the line
regex$	Must match at the end of the line
[abc]	Set definition, matches the letter a or b or c
[abc][vz]	Set definition, matches a or b or c followed by v or z
[^abc]	Negates the pattern. Matches any character except a or b or c
[a-d1-7]	Ranges, letter between a and d or digits from 1 to 7, will not match d1
X|Z	Finds X or Z
XZ	Finds X directly followed by Z
\d	Any digit, short for [0-9]
\D	A non-digit, short for [^0-9]
\s	A whitespace character, short for [\t\n\x0b\r\f]
\S	A non-whitespace character, for short

Active Code Completion with GRAPHITE


```
import java.util.regex.Pattern;

public class Matcher {
    public static boolean isTemperature(String s) {
        Pattern p =
    }
}
```

^\d\d$

Ignore Case

Should match...	Should NOT match...
37F	12:05
42.1 F	37
.8C	37Q
-10C	

= matched by pattern

Pattern	Description
.	Matches any character
^regex	Must match at the beginning of the line
regex$	Must match at the end of the line
[abc]	Set definition, matches the letter a or b or c
[abc][vz]	Set definition, matches a or b or c followed by v or z
[^abc]	Negates the pattern. Matches any character except a or b or c
[a-d1-7]	Ranges, letter between a and d or digits from 1 to 7, will not match d1
X|Z	Finds X or Z
XZ	Finds X directly followed by Z
\d	Any digit, short for [0-9]
\D	A non-digit, short for [^0-9]
\s	A whitespace character, short for [\t\n\x0b\r\f]
\S	A non-whitespace character, for short

Active Code Completion with GRAPHITE


```
import java.util.regex.Pattern;

public class Matcher {
    public static boolean isTemperature(String s) {
        Pattern p =
    }
}
```

^\d+(\.\d+)?\s?(F|C)$

Ignore Case

Should match...	Should NOT match...
37F	12:05
42.1 F	37
.8C	37Q
-10C	

= matched by pattern

Pattern	Description
.	Matches any character
^regex	Must match at the beginning of the line
regex$	Must match at the end of the line
[abc]	Set definition, matches the letter a or b or c
[abc][vz]	Set definition, matches a or b or c followed by v or z
[^abc]	Negates the pattern. Matches any character except a or b or c
[a-d1-7]	Ranges, letter between a and d or digits from 1 to 7, will not match d1
X|Z	Finds X or Z
XZ	Finds X directly followed by Z
\d	Any digit, short for [0-9]
\D	A non-digit, short for [^0-9]
\s	A whitespace character, short for [\t\n\x0b\r\f]
\S	A non-whitespace character, for short

Active Code Completion with GRAPHITE


```
import java.util.regex.Pattern;

public class Matcher {
    public static boolean isTemperature(String s) {
        Pattern p =
    }
}
```

^-?(\d+|(\d*(\.\d+)?\s?(F|C)$

☐ Ignore Case

Should match...	Should NOT match...
37F	12:05
42.1 F	37
.8C	37Q
-10C	

= matched by pattern

Pattern	Description
.	Matches any character
^regex	Must match at the beginning of the line
regex$	Must match at the end of the line
[abc]	Set definition, matches the letter a or b or c
[abc][vz]	Set definition, matches a or b or c followed by v or z
[^abc]	Negates the pattern. Matches any character except a or b or c
[a-d1-7]	Ranges, letter between a and d or digits from 1 to 7, will not match d1
X|Z	Finds X or Z
XZ	Finds X directly followed by Z
\d	Any digit, short for [0-9]
\D	A non-digit, short for [^0-9]
\s	A whitespace character, short for [\t\n\x0b\r\f]
\S	A non-whitespace character, for short

Active Code Completion with GRAPHITE


```
import java.util.regex.Pattern;

public class Matcher {
    public static boolean isTemperature(String s) {
        Pattern p =
    }
}
```

^-?(\d+|(\d*(\.\d+)))?\s?(F|C)$

☐ Ignore Case

Should match...	Should NOT match...
37F	12:05
42.1 F	37
.8C	37Q
-10C	

= matched by pattern

Pattern	Description
.	Matches any character
^regex	Must match at the beginning of the line
regex$	Must match at the end of the line
[abc]	Set definition, matches the letter a or b or c
[abc][vz]	Set definition, matches a or b or c followed by v or z
[^abc]	Negates the pattern. Matches any character except a or b or c
[a-d1-7]	Ranges, letter between a and d or digits from 1 to 7, will not match d1
X|Z	Finds X or Z
XZ	Finds X directly followed by Z
\d	Any digit, short for [0-9]
\D	A non-digit, short for [^0-9]
\s	A whitespace character, short for [\t\n\x0b\r\f]
\S	A non-whitespace character, for short

Active Code Completion with GRAPHITE


```java
import java.util.regex.Pattern;

public class Matcher {
    public static boolean isTemperature(String s) {
        Pattern p = Pattern.compile("^-?(\\d+|(\\d*(\\.\\d+)))?\\s?(F|C)$");
        /*
         * Should match:
         *   37F
         *   42.1 F
         *   .8C
         *   -10C
         *
         * Should NOT match:
         *   12:05
         *   37
         *   37Q
         *
         */

    }
}
```

Our Design Methodology

1. Large developer survey to validate this idea and develop **design criteria** and **use cases** *before implementation!*

2. Tool design and implementation (GRAPHITE)

3. Controlled pilot study to justify usefulness claims

Survey Method

- Target: professional developers
- Participants recruited from
  - “reddit.com” programming forum
    - ~340,000 registered readers
  - Local CS graduate students mailing list (22)
- 696 people started the survey (~20 minutes long)
  - **475** people completed the survey, we only analyzed these responses

Participant Experience

- Participant's experience with regular expressions and SQL

	Regular Expressions	SQL
Never used	4.8%	0.0%
Use infrequently	46.7%	37.4%
Use frequently	**48.4%**	**62.6%**

- Along with experience with programming languages, implies that **most participants are professional programmers**

I. Mockups

- For each of three classes (**Color**, **Regular Expressions**, **SQL query**)
  - Ask which strategy they would naturally use to instantiate the given class
  - Show them mockup screenshots of our tool
  - Ask how often they would use the tool if they wanted to instantiate the class
  - *Ask them to qualify the answer or make suggestions* **(open-ended)**

Mockup – Color

*ColorTest.java

```
1  import java.awt.Color;
2
3
4  public class ColorTest {
5
6      public static void main(String[] args) {
7          Color color =
8      }
9
10 }
11
```

Show interactive palette - Color
color : Color
args : String []
main(String[] args) : void - ColorTest
ColorTest

Create a color using a color palette.

Press '^Space' to show Template Proposals

Press 'Tab' from proposal table or click for focus

Outline
import declarations
ColorTest
main(String[]) : void

Mockup – Color

*ColorTest.java

```
1  import java.awt.Color;
2
3
4  public class ColorTest {
5
6      public static void main(String[] args) {
7          Color color =
8      }
9
10 }
11
```

RGB
R: 0
G: 148
B: 255
Hex: 0094FF
HSV
H: 205
S: 100
V: 100
Transparency - Alpha
255

Outline
import declarations
ColorTest
main(String[]) : voic

Mockup – Color

*ColorTest.java

```
1  import java.awt.Color;
2
3
4  public class ColorTest {
5
6      public static void main(String[] args) {
7          Color color = new Color(0, 148, 255, 255);
8      }
9
10 }
11
```

Outline

import declarations
ColorTest
main(String[]) : voic

Mockup – Regular Expressions

*EmailVerifier.java

```
package examples;

import java.util.regex.*;

public class EmailVerifier {
    public static boolean verify(String s) {
        Pattern emailPattern =

    }
}
```

\w+@\w+\.[(edu)|(org)|(net)|(com)]

add positive | negative Flags: i m s u

+ albert@princeton.edu ✔
+ hawking@cam.ac.uk ✘
- jim@yahoo ✔

Regex Reference

Outline

- examples
- import declarations
  - java.util.regex.*
- EmailVerifier
  - verify(String) : boolean

Mockup – SQL queries

JdbcTest.java

```
import java.sql.*;

class JdbcTest {
    public static void main(String[] args) {
        ResultSet custTowns =
```

SELECT CustomerTown FROM Persons

Connect to... Send Query

CustomerTown
Lincoln
Manchester
Nottingham
Nottingham
Swindon
London
London
Wigan

I. Mockups

- For each of three classes (**Color**, **Regular Expressions**, **SQL query**)
  - **Ask which strategy they would naturally use to instantiate the given class**
  - Show them mockup screenshots of our tool
  - Ask how often they would use the tool if they wanted to instantiate the class
  - *Ask them to qualify the answer or make suggestions* **(open-ended)**

Default Strategy – Regex, SQL

	Regular Expressions	SQL
Separate test script	29.7%	15.5%
Guess and check	13.4%	16.0%
External tool	**38.5%**	**58.9%**
Search for examples	12.1%	4.8%
Other	6.2%	4.8%

I. Mockups

- For each of three classes (**Color**, **Regular Expressions**, **SQL query**)
  - Ask which strategy they would naturally use to instantiate the given class
  - Show them mockup screenshots of our tool
  - **Ask how often they would use the tool if they wanted to instantiate the class**
  - *Ask them to qualify the answer or make suggestions* **(open-ended)**

Usefulness of Mockups

- "Consider situations where you need to instantiate the [specified] class. What portion of the time, in these situations, do you think you would use this feature?"

CLASS	Nearly every time	Most of the time	Some of the time	Rarely	Never
Color	9.6%	22.1%	**32.4%**	28.2%	7.7%
RegExp	**36.6%**	29.5%	21.8%	7.3%	4.8%
SQL	18.2%	19.3%	**30.9%**	20.4%	11.4%

I. Mockups

- For each of three classes (**Color**, **Regular Expressions**, **SQL query**)
  - Ask which strategy they would naturally use to instantiate the given class
  - Show them mockup screenshots of our tool
  - Ask how often they would use the tool if they wanted to instantiate the class
  - ***Ask them to qualify the answer or make suggestions* (open-ended)**

System Design Constraints

- Reversibility (**19** across classes)
  - Bring palette back up from data
- Palette settings and state
  - Wanted recent regexes + control over comments (**12**)
  - Persistent database connection information (**9**)
  - Recent colors (**20**)
- IDE/language independence
  - Several expressed a desire for IDE or even language independence of this feature

Active Code Completion with GRAPHITE


```java
import java.util.regex.Pattern;

public class Matcher {
    public static boolean isTemperature(String s) {
        Pattern p = Pattern.compile("^-?(\\d+|(\\d*(\\.\\d+)))?\\s?(F|C)$");
        /*
         * Should match:
         *   37F
         *   42.1 F
         *   .8C
         *   -10C
         *
         * Should NOT match:
         *   12:05
         *   37
         *   37Q
         *
         */

    }
}
```

Active Code Completion with GRAPHITE


```
import java.util.regex.Pattern;

public class Matcher {
    public static boolean isTemperature(String s) {
        Pattern p = Pattern.compile("^-?(\\d+|(\\d*(\\.\\d+)))?\\s?(F|C)$");
        /*
         * Should ma
         *   37F
         *   42.1 F
         *   .8C
         *   -10C
         *
         * Should NO
         *   12:05
         *   37
         *   37Q
         *
         */

    }
}
```

Use the regular expression workbench...
Pattern – java.util.regex
p : Pattern
s : String
isTemperature(String s) : boolean – Match
Matcher – edu.cmu.cs.comar
runnable – runnable

Press '^Space' to show Template Proposals

Displays a workbench that allows you to enter a regular expression pattern and test it against positive and negative examples. Automatically handles escape sequences!

Press 'Tab' from proposal table or click for focus

Active Code Completion with GRAPHITE


```
import java.util.regex.Pattern;

public class Matcher {
    public static boolean isTemperature(String s) {
        Pattern p = Pattern.compile("^-?(\\d+|(\\d*(\\.\\d+)))?\\s?(F|C)$");
        /*
         * Should ma
         *   37F
         *   42.1 F
         *   .8C
         *   -10C
         *
         * Should NO
         *   12:05
         *   37
         *   37Q
         *
         */

    }
}
```

^-?(\d+|(\d*(\.\d+)))?\s?(F|C)$

☐ Ignore Case

Should match...	Should NOT match...
37F	12:05
42.1 F	37
.8C	37Q
-10C	

= matched by pattern

Pattern	Description
.	Matches any character
^regex	Must match at the beginning of the line
regex$	Must match at the end of the line
[abc]	Set definition, matches the letter a or b or c
[abc][vz]	Set definition, matches a or b or c followed by v or z
[^abc]	Negates the pattern. Matches any character except a or b or c
[a-d1-7]	Ranges, letter between a and d or digits from 1 to 7, will not match d1
X|Z	Finds X or Z
XZ	Finds X directly followed by Z
\d	Any digit, short for [0-9]
\D	A non-digit, short for [^0-9]
\s	A whitespace character, short for [\t\n\x0b\r\f]
\S	A non-whitespace character, for short

System Design Constraints

- Reversibility (**19** across classes)
  - Bring palette back up from data
- Palette settings and state
  - Wanted recent regexes + control over comments (**12**)
  - Persistent database connection information (**9**)
  - Recent colors (**20**)
- IDE/language independence
  - Several expressed a desire for IDE or even language independence of this feature

Graphite: Regular Expression

file:///Users/cyrus/Dropbox/projects/graphite/graphite/palettes/regex/regex.html

Social | Music | Checkvists | Awesum | Tools | Other Bookmarks

^\d\d(C|F)$

Ignore Case

Should match...

37F

37 F

Should NOT match...

*(enter **negative** test cases above, pressing ENTER between each one)*

= matched by pattern

Pattern	Description
.	Matches any character
^regex	Must match at the beginning of the line
regex$	Must match at the end of the line
[abc]	Set definition, matches the letter a or b or c
[abc][vz]	Set definition, matches a or b or c followed by v or z
[^abc]	Negates the pattern. Matches any character except a or b or c
[a-d1-7]	Ranges, letter between a and d or digits from 1 to 7, will not match d1
X|Z	Finds X or Z
XZ	Finds X directly followed by Z
\d	Any digit, short for [0-9]
\D	A non-digit, short for [^0-9]

Elements | Resources | Network | Scripts | Timeline | Profiles | Audits | Search Scripts

regex.js

```
24          var height = t[0].offsetHeight + 0;
25          console.log(height);
26          if (height < minHeight) height = minHeight;
27          graphite.resizeTo(width, height);
28      };
29
30      // Entry behavior
31      var ENTER = 13;
32
33      var waiting = false; // have to do it this way to deal with weird bugs
34      entry.keydown(function(e) {
35          if (e.keyCode == ENTER) { waiting = true; }
36      });
37
38      entry.keyup(function(e) {
39          runTests();
40
41          if (waiting) {
42                  waiting = false;
43                  enter();
44          }
```

Watch Expressions

Call Stack

Not Paused

Scope Variables

Not Paused

Breakpoints

No Breakpoints

DOM Breakpoints

XHR Breakpoints

Event Listener Breakpoints

Workers

186

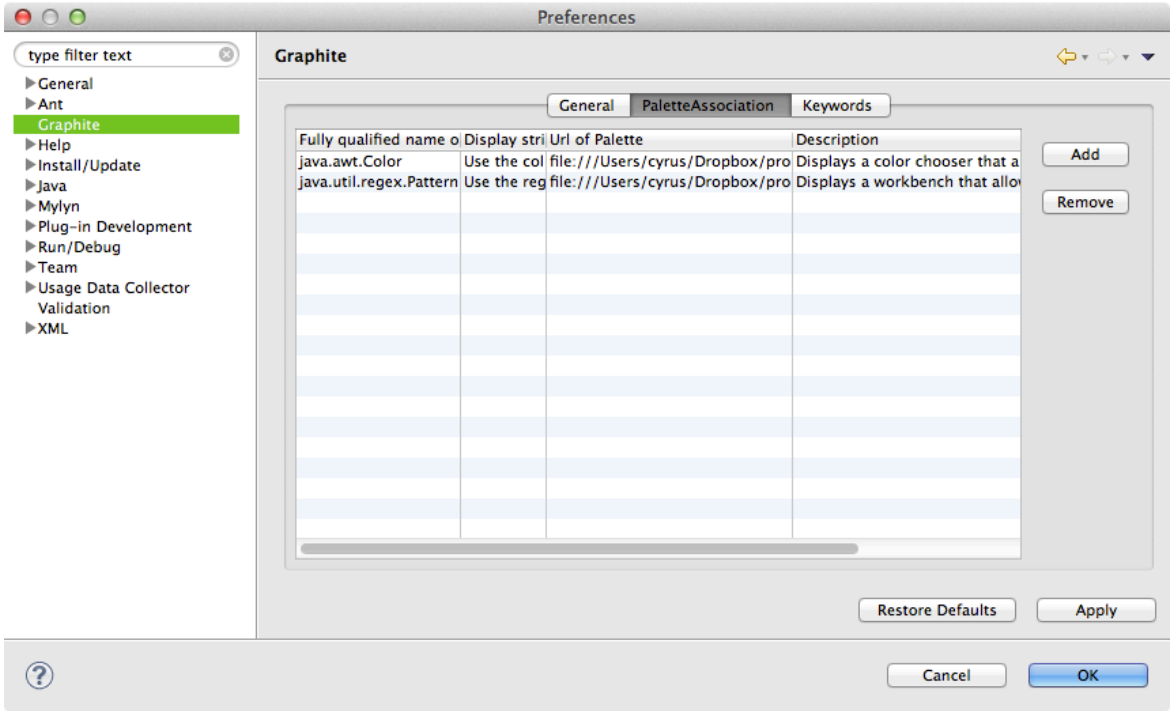
Associating a Palette with a Class

```
import edu.cmu.cs.graphite.GraphitePalette;

@GraphitePalette(url="http://www.cs.cmu.edu/~comar/regex.html",
    description="Regular expression matcher!")
public class Pattern {
    // ...

}
```

^ or ->



II. Suggestions

- Ask what other classes could benefit from these kind of interactive palettes **(open-ended)**

Suggestions

- Alternatives / Tricky / Literal Syntax (16)
  - Example classes
    - string, vector, dictionary, matrix
    - URLs, paths
    - embedded languages (e.g. HTML)

Collection Class	Total	Literal	Percentage
`ArrayList`	464	44	9.5%
`HashMap`	56	19	33.9%
`HashSet`	122	62	50.8%
`Hashtable`	86	10	11.6%
`Vector`	729	31	4.2%
Total	1457	166	11.4%

Figure 4. Usage patterns for common Java collection classes in the `java.util` package in our code corpus. Uses that fit a pattern that can be captured by a literal make up a significant portion of all uses. Not all possible usage scenarios of this type were captured by our analysis, so these numbers are lower bounds.

Suggestions

- Unclear Parameter Implications (11)
  - Example classes
    - audio tweaks
    - 3D transformation matrices
    - number/string/date formatting
  - What people wanted
    - modify the parameters and see the results directly without running the application

Suggestions

- Query Languages (17)
  - Example classes
    - regular expressions
    - SQL queries
    - XPath / XQuery
  - What people wanted
    - testing the query result directly without running the application

Suggestions

- Graphical Elements (27)
  - What people wanted
    - checking the graphical property directly
    - manipulating the layout of GUI elements
  - Example classes
    - color, font, shape, thickness, etc.
    - JFrame, layouts, any swing components, etc.

Suggestions

- Describe by Example (2)
  - Example classes
    - keyboard keys (e.g., shortcut keys)
- Integrating with Documentation, Tutorials (7)

Pilot Study Overview

- Observe users doing **regular expression** tasks with and without Graphite.
- Hypotheses:
  - Control subjects (4) will have more trouble with:
    - Factory pattern initialization: `Pattern.compile(…)` [Ellis et al, ICSE07]
    - Escaping of strings: `Pattern.compile("\\d")`
  - Treatment subjects (3) will complete more tasks and test more examples

Protocol

- Pre-survey asking the familiarity with:
  - Programming languages
  - Integrated development environments
  - Regular expressions
- Give a demo of how to use Graphite plug-in with the Color palette
  - Only for the treatment group
- Instruct them how to complete the tasks and start the experiment

Pilot Study Results

- Hypotheses:
  - ✓ Control subjects had trouble with:
    - ✓ Factory pattern initialization: `Pattern.compile(…)` [Ellis et al, ICSE07]
    - ✓ Escaping of strings: `Pattern.compile("\\d")`
  - ✓ Treatment subjects completed more tasks and tested more examples

Pilot Study Results

- Hypotheses:
  - ✓ Control subjects had trouble with:
    - ✓ Factory pattern initialization: `Pattern.compile(…)` [Ellis et al, ICSE07]
    - ✓ Escaping of strings: `Pattern.compile("\\d")`
  - ✓ Treatment subjects completed more tasks and tested more examples
- Treatment subjects were uniformly positive. Control subjects later shown the palette were also positive.

DEFINITION. Active Code Completion

- A code completion system that **actively engages** both users and providers during code completion
  1. Providers equip types with specialized user interfaces (*palettes*)
  2. API clients interact with palettes to provide requested information about their intent
  3. API providers generate code based on this information

Benefits of Active Code Completion

- Domain-specific tools can be easily **integrated** directly into the editor (fewer context switches) and **discovered** more easily.

- Palettes can handle **tricky aspects of an API** (e.g. factory patterns, string escaping)

Our Design Methodology

1. Large developer survey to validate this idea and develop **design criteria** and **use cases** *before implementation!*

2. Tool design and implementation (GRAPHITE) http://www.cs.cmu.edu/~NatProg/graphite.html

3. Controlled pilot study to justify usefulness claims

Broader Trends

Extensible languages require extensible development environments.

Active typing: directly equipping types with relevant compile-time and design-time logic.

More to come!

Thanks!

- Jonathan Aldrich and the CMU PLAID group
- Students in 05-899D: Human Aspects of Software Development
  (co-taught by Thomas LaToza and Brad Myers)
- UIUC Software Engineering Seminar
- All our participants and subjects!