How we automate code maintenance and you can too!

gitar

Gautam Korlam
Ameya Ketkar

About us

Gautam Korlam

- Co-Founder at Gitar
- Previously, Principal Engineer on Uber's Developer Platform Team for 10 years
- Passion - tools to improve developer experience

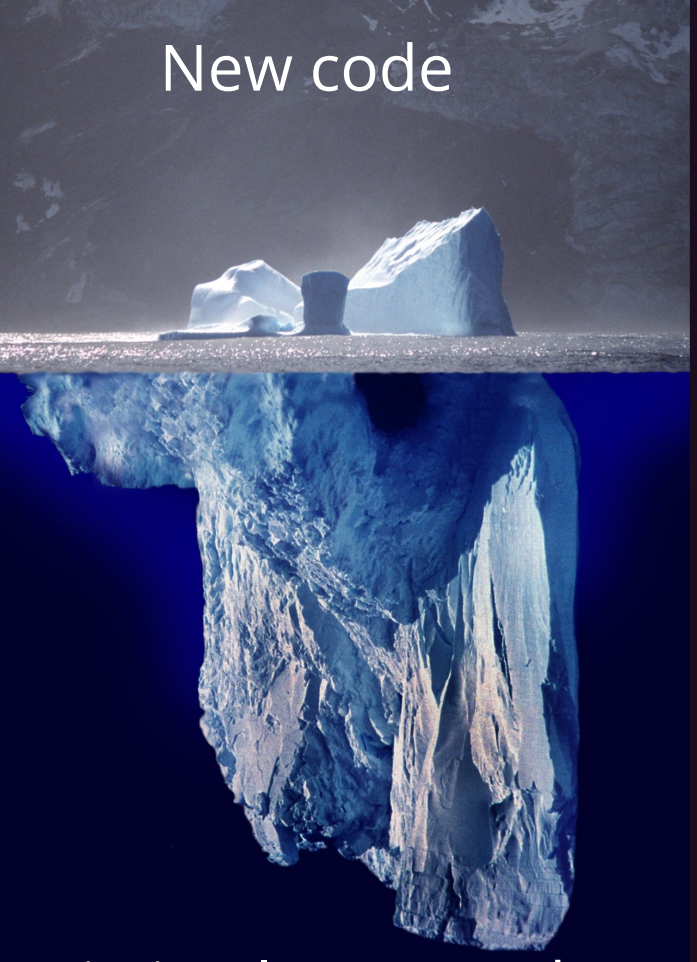
Ameya Ketkar

- Founding Engineer at Gitar
- Deleted and migrated over **500K LOC**
- PhD from Oregon State University in program analysis and code rewriting

Key Takeaways

- Code maintenance takes too much time
- Automation is key ... but is hard
- Gitar automates code maintenance

gitar.co



Developers spend 30% of time on code maintenance

- Patching Security Vulnerabilities
- Dependency Management
- Removing Dead Code
- Removing Duplicate Code
- Cleaning Up Stale Feature Flags
- Performance Profiling & Optimization
- Implementing Best Practices
- Code Review Fixes

Manual code maintenance is unsustainable

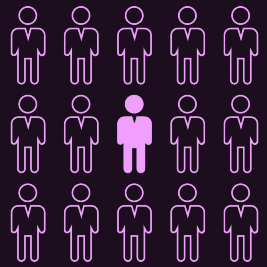
Today’s Reality

Efficiency

There is a push to do more with less

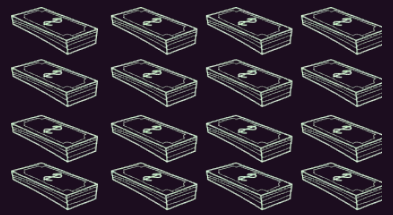
Lean teams

Small platform teams enable large product teams



Automation

Automation is key for 10x lift



Today's automation: Insufficient

- At best they automate ticket assignment
- Not very well integrated into existing tools
- Not cross language

A hard problem

Reliability

Accuracy

Coverage

Performance

Why is it hard to automate **migrations**?

Junit 4 to 5

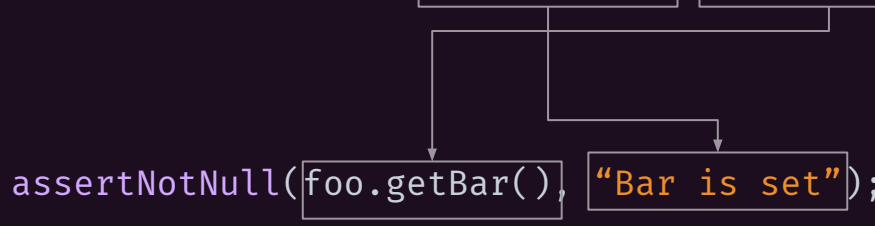
```
import org.junit.Test;

@RunWith(MockitoJUnitRunner.class)
public class FooTest {
   @BeforeClass
   public void setUp() {
     …
   }

   @Test
   public void someTest() {
     …
   }

   @Test
   @Ignore
   public void someOtherTest() {
     …
   }
}
```

```
import org.junit.jupiter.api.Test;

@ExtendWith(MockitoExtension.class)
public class FooTest {
   @BeforeAll
   public void setUp() {
     …
   }

   @Test
   public void someTest() {
     …
   }

   @Test
   @Disabled
   public void someOtherTest() {
     …
   }
}
```

```
Assert.assertNotNull(“Bar is set”, foo.getBar());
```

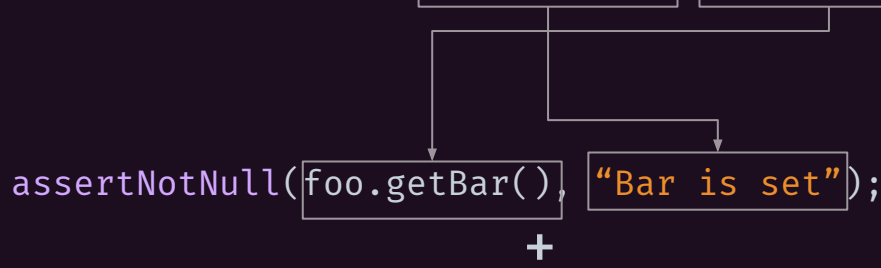
```
 assertNotNull(foo.getBar(), “Bar is set”);
```

```
Assert.assertNotNull("Bar is set", foo.getBar());
```



```
assertNotNull(foo.getBar(), "Bar is set");
```

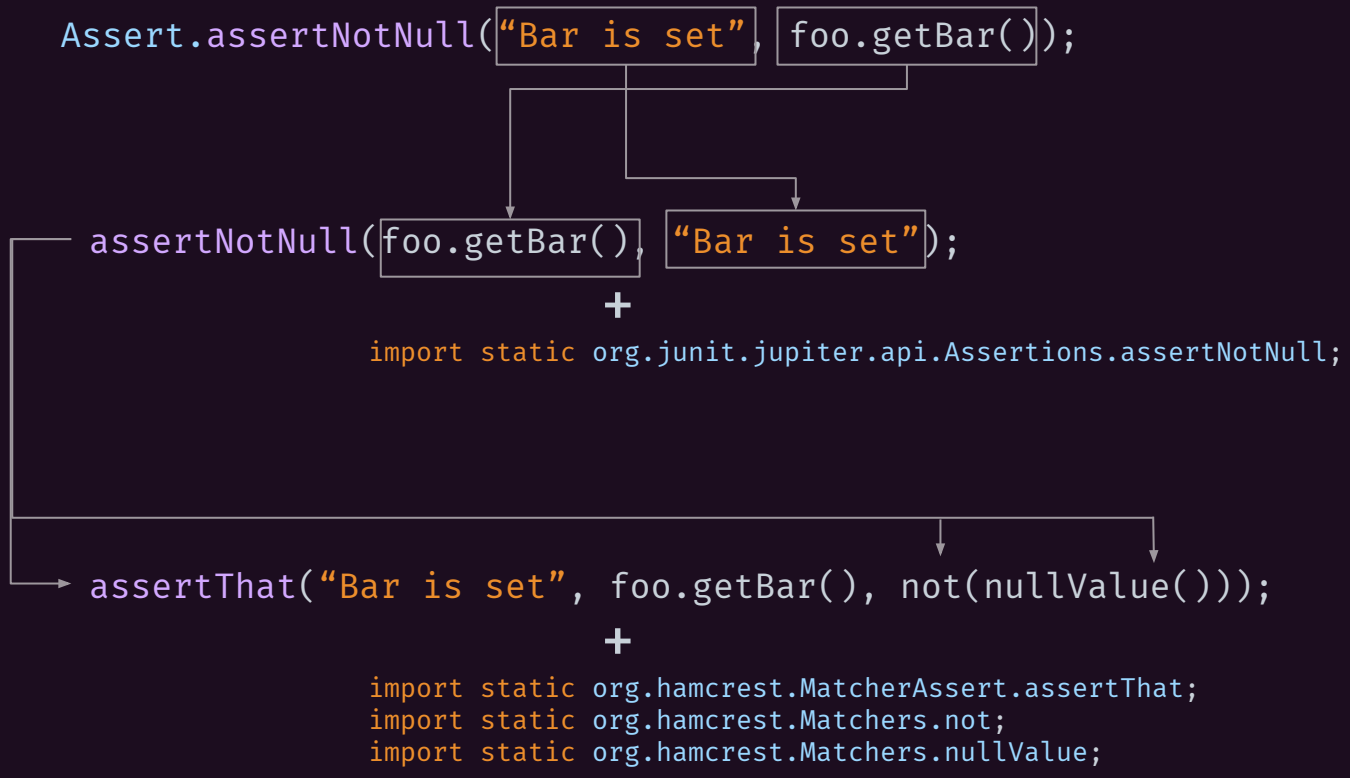
```
Assert.assertNotNull("Bar is set", foo.getBar());
```



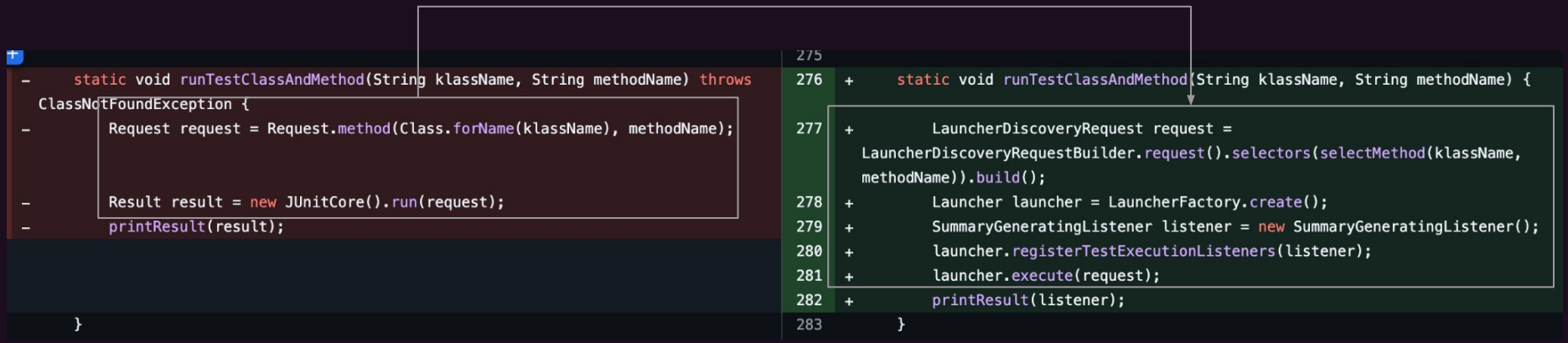
```
assertNotNull(foo.getBar(), "Bar is set");
```

+

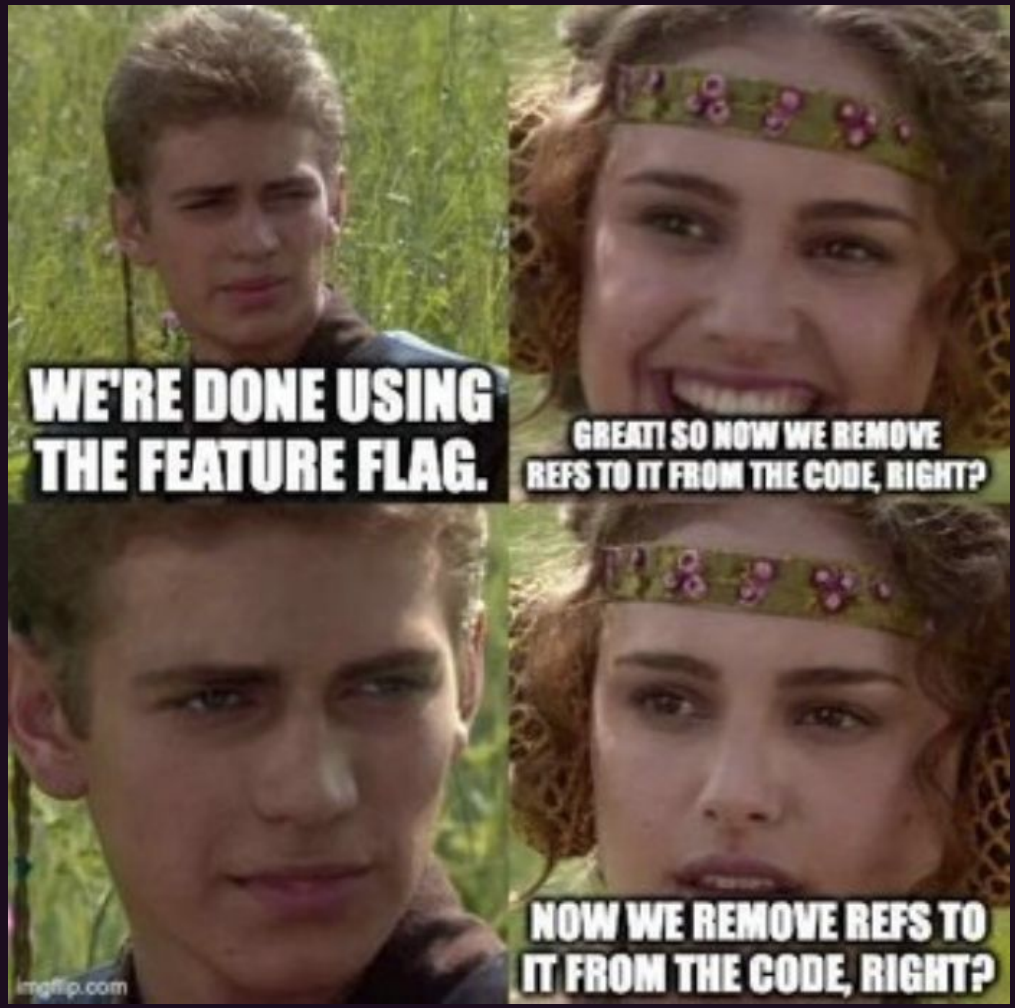
```
import static org.junit.jupiter.api.Assertions.assertNotNull;
```


Assert.assertNotNull("Bar is set", foo.getBar());
assertNotNull(foo.getBar(), "Bar is set");
+
import static org.junit.jupiter.api.Assertions.assertNotNull;
assertThat("Bar is set", foo.getBar(), not(nullValue()));
+
import static org.hamcrest.MatcherAssert.assertThat;
import static org.hamcrest.Matchers.not;
import static org.hamcrest.Matchers.nullValue;


@Test(expected = SomeException.class)
public void someTest() {
x = Foo.bar();
Foo.baz(x, 54);
}
@Test
public void someTest() {
assertThrows(
SomeException.class,
() -> {
x = Foo.bar();
Foo.baz(x, 54);
});
}


- static void runTestClassAndMethod(String klassName, String methodName) throws ClassNotFoundException {
- Request request = Request.method(Class.forName(klassName), methodName);
- Result result = new JUnitCore().run(request);
- printResult(result);
}
275
276 + static void runTestClassAndMethod(String klassName, String methodName) {
277 + LauncherDiscoveryRequest request = LauncherDiscoveryRequestBuilder.request().selectors(selectMethod(klassName, methodName)).build();
278 + Launcher launcher = LauncherFactory.create();
279 + SummaryGeneratingListener listener = new SummaryGeneratingListener();
280 + launcher.registerTestExecutionListeners(listener);
281 + launcher.execute(request);
282 + printResult(listener);
283 }

Why is it hard to automate
dead code cleanup?


WE'RE DONE USING THE FEATURE FLAG.
GREAT! SO NOW WE REMOVE REFS TO IT FROM THE CODE, RIGHT?
NOW WE REMOVE REFS TO IT FROM THE CODE, RIGHT?
imgflip.com

Stale Feature Flag Cleanup

```
public interface Parameter {
    @Flag(name = "new-banner") boolean isNewBanner()
    …
}
```

Stale feature flag

```
fun showBanner() {
  if (uiParameter.isNewBanner() || checkUser()) {
    newBanner()
    return
  } else {
    oldBanner()
  }
  println("Old banner is displayed")
  return
}
fun oldBanner() { ... }
fun newBanner() { ... }
```

Stale Feature Flag Cleanup

```
public interface Parameter {
    @Flag(name = "new-banner") boolean isNewBanner()
    …
}
```

Stale feature flag

```
fun showBanner() {
  if (uiParameter.isNewBanner() || checkUser()) {
    newBanner()
    return
  } else {
    oldBanner()
  }
  println("Old banner is displayed")
  return
}
fun oldBanner() { ... }
fun newBanner() { ... }
```

Delete new_dashboard declaration

```java
public interface Parameter {
   @Flag(name = "new-banner") boolean isNewBanner()
   …
}
```

```kotlin
fun showBanner() {
  if (uiParameter.isNewBanner() || checkUser()) {
    newBanner()
    return
  } else {
    oldBanner()
  }
  println("Old banner is displayed")
  return
}
fun oldBanner() { ... }
fun newBanner() { ... }
```

Replace check with true

```
public interface Parameter {
   …
}
```

```
fun showBanner() {
  if (uiParameter.isNewBanner() || checkUser()) {
    newBanner()
    return
  } else {
    oldBanner()
  }
  println("Old banner is displayed")
  return
}
fun oldBanner() { ... }
fun newBanner() { ... }
```

Simplify disjunction

```
public interface Parameter {
    …
}
```

```
fun showBanner() {
  if (true || checkUser()) {
    newBanner()
    return
  } else {
    oldBanner()
  }
  println("Old banner is displayed")
  return
}
fun oldBanner() { ... }
fun newBanner() { ... }
```

Simplify control-flow

```
public interface Parameter {
    …
}
```

```
fun showBanner() {
  if (true) {
    newBanner()
    return
  } else {
    oldBanner()
  }
  println("Old banner is displayed")
  return
}
fun oldBanner() { ... }
fun newBanner() { ... }
```

Delete unreachable statements

```
public interface Parameter {
    …
}
```

```
fun showBanner() {
  newBanner()
  return
  println("Old banner is displayed")
  return
}
fun oldBanner() { ... }
fun newBanner() { ... }
```

Delete unused code

```
public interface Parameter {
    …
}
```

```
fun showBanner() {
  newBanner()
  return
}
fun oldBanner() { ... }
fun newBanner() { ... }
```

Am I to morph into a compiler?

```
public interface Parameter {
    …
}
```

```
fun showBanner() {
  newBanner()
  return
}

fun newBanner() { ... }
```

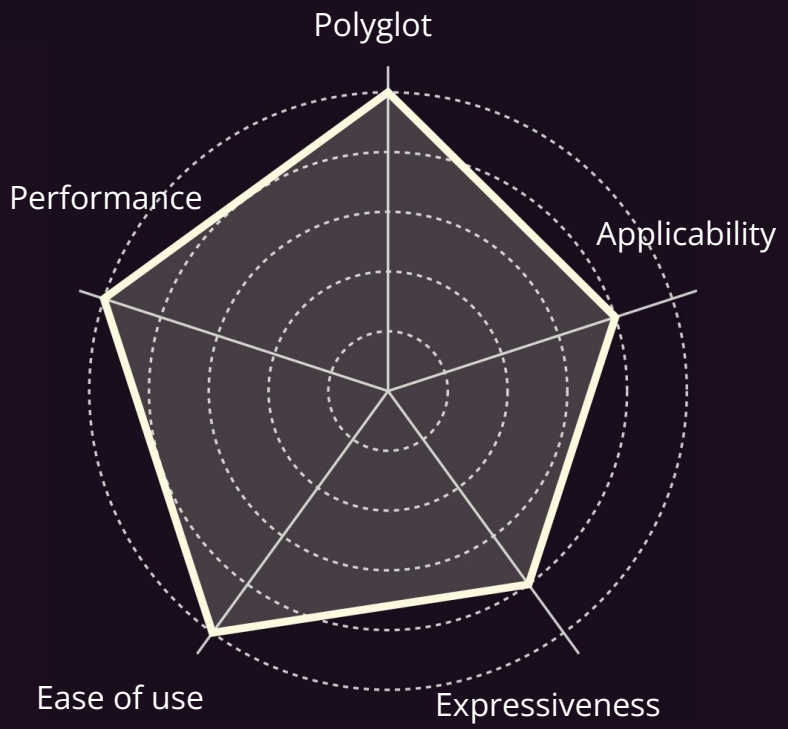
Rinse and repeat across platforms





How Gitar automates code maintenance

A novel framework



github.com/uber/piranha

We created Polyglot Piranha

We actively maintain it

Read our paper:

A Lightweight Polyglot Code Transformation Language

ACM Conference on Programming Language Design & Implementation, 2024

https://gitar.co/polyglot-piranha-paper

gitar

1800

Merged PRs

> 90%

No additional changes

500K

Lines of Code

6

months

Equivalent to a team of 29 devs

Gitar in action!

[Gitar] Migrating from JUnit4 to JUnit5 #23

Edit | Code

Open **gitar-bot** wants to merge 2 commits into `master` from `gitar_migration_demo_only_junit4_to_junit5_1727289510`

Conversation 0 | Commits 2 | Checks 0 | Files changed 105 | +2,313 −1,879

Changes from all commits | File filter | Conversations | 0 / 105 files viewed | Review in codespace | Review changes

16 pom.xml | Viewed

```
@@ -2039,9 +2039,19 @@
```

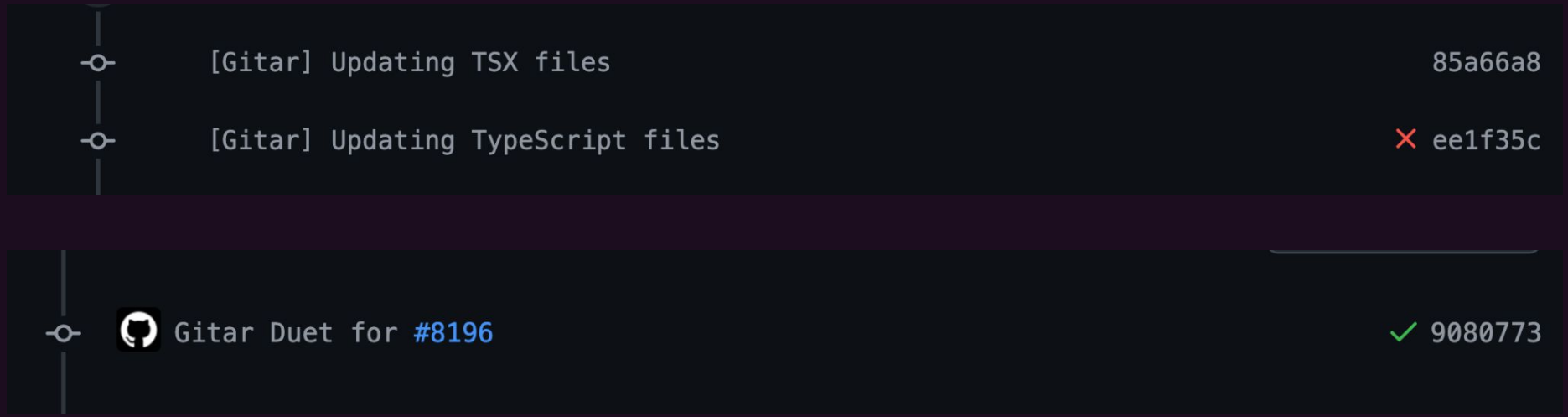
```
2039                 <version>10.12.3</version>
2040             </dependency>
2041             <dependency>
2042 -               <groupId>junit</groupId>
2043 -               <artifactId>junit</artifactId>
2044 -               <version>4.13.2</version>
```

```
2039                 <version>10.12.3</version>
2040             </dependency>
2041             <dependency>
2042 +               <groupId>org.junit.jupiter</groupId>
2043 +               <artifactId>junit-jupiter</artifactId>
2044 +               <version>5.11.0</version>
2045 +           </dependency>
2046 +           <dependency>
2047 +               <groupId>org.junit.platform</groupId>
2048 +               <artifactId>junit-platform-suite</artifactId>
2049 +               <version>1.11.0</version>
2050 +           </dependency>
2051 +           <dependency>
2052 +               <groupId>org.hamcrest</groupId>
2053 +               <artifactId>hamcrest</artifactId>
2054 +               <version>3.0</version>
```

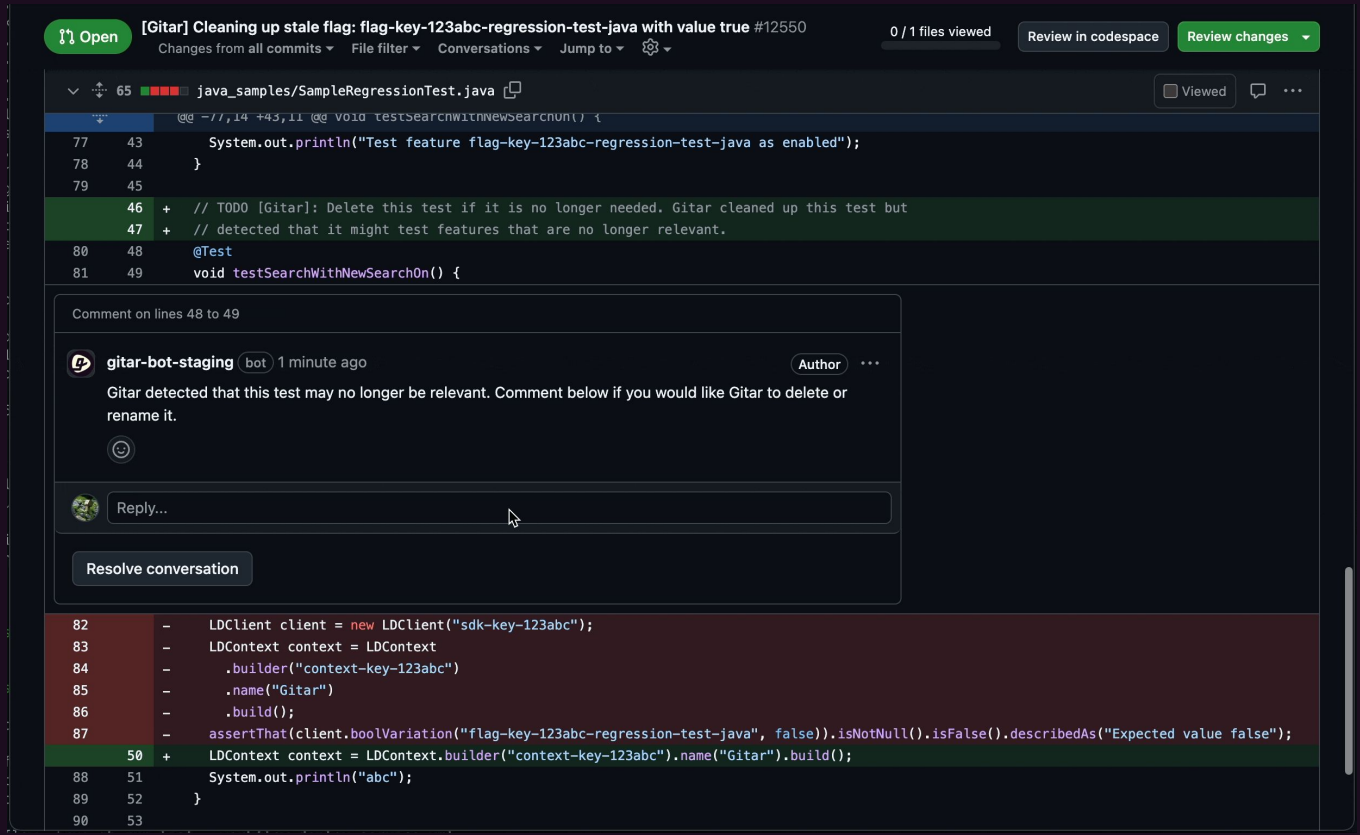
```
@@ -28,6 +28,12 @@
28  import java.util.ArrayList;
29  import java.util.Date;
30  import java.util.HashSet;
31 + import org.junit.platform.launcher.Launcher;
32 + import org.junit.platform.launcher.LauncherDiscoveryRequest;
33 + import org.junit.platform.launcher.core.LauncherDiscoveryRequestBuilder;
34 + import org.junit.platform.launcher.core.LauncherFactory;
35 + import org.junit.platform.launcher.listeners.SummaryGeneratingListener;
36 + import org.junit.platform.launcher.listeners.TestExecutionSummary;
31  import org.junit.runner.JUnitCore;
32  import org.junit.runner.Request;
33  import org.junit.runner.Result;
@@ -48,8 +54,12 @@ public class TornadoHelper {
48    public static final boolean OPTIMIZE_LOAD_STORE_SPIRV =
49  Boolean.parseBoolean(System.getProperty("tornado.spirv.loadstore", "False"));
50
51 -   private static void printResult(Result result) {
52 -     System.out.printf("Test ran: %s, Failed: %s%n", result.getRunCount(), result.getFailureCount());
```

```
@@ -28,6 +28,12 @@
28  import java.util.ArrayList;
29  import java.util.Date;
30  import java.util.HashSet;
31 + import org.junit.platform.launcher.Launcher;
32 + import org.junit.platform.launcher.LauncherDiscoveryRequest;
33 + import org.junit.platform.launcher.core.LauncherDiscoveryRequestBuilder;
34 + import org.junit.platform.launcher.core.LauncherFactory;
35 + import org.junit.platform.launcher.listeners.SummaryGeneratingListener;
36 + import org.junit.platform.launcher.listeners.TestExecutionSummary;
37  import org.junit.runner.JUnitCore;
38  import org.junit.runner.Request;
39  import org.junit.runner.Result;
@@ -54,8 +60,12 @@ public class TornadoHelper {
54    public static final boolean OPTIMIZE_LOAD_STORE_SPIRV =
55  Boolean.parseBoolean(System.getProperty("tornado.spirv.loadstore", "False"));
56
57 +   // Gitar's advanced AI and code analysis agent changed this method
58 +   // AI enhanced changes are currently in beta.
```

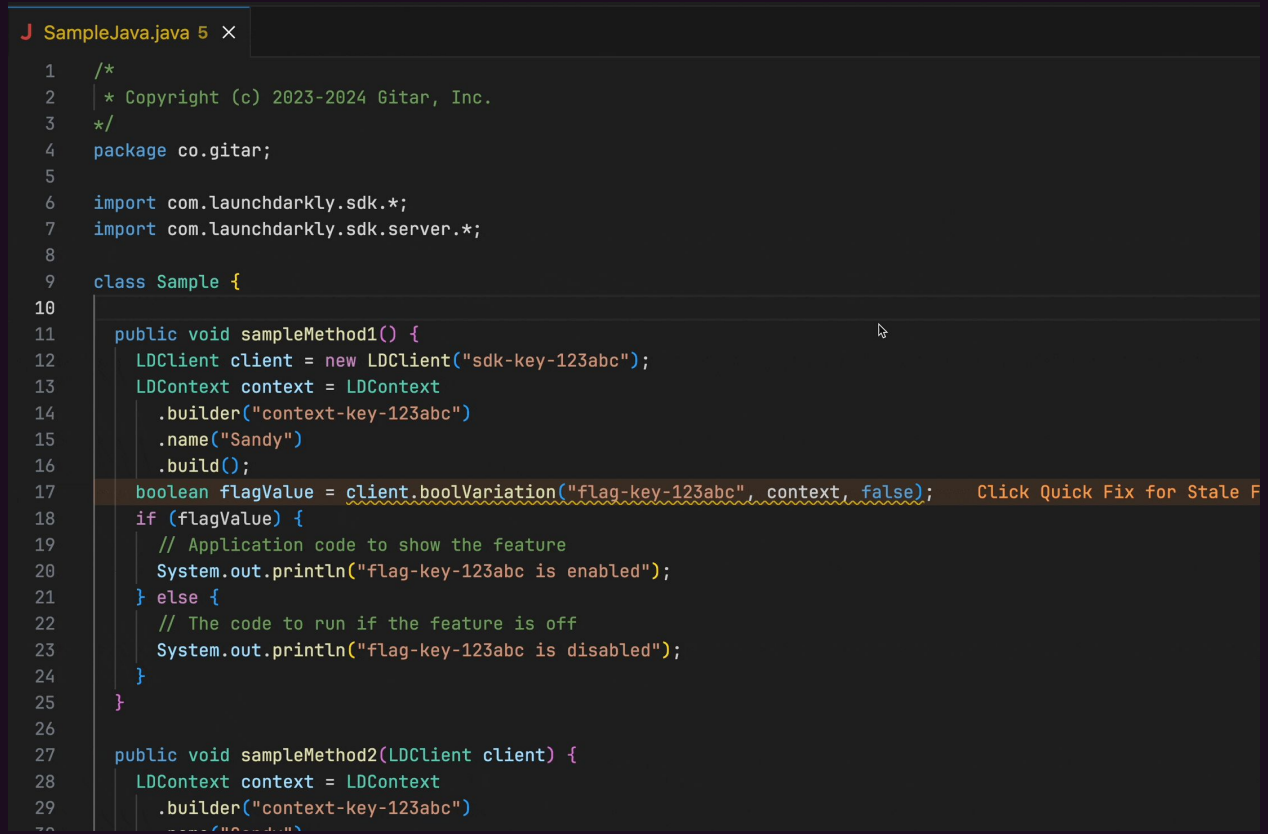
Autofix build errors



Interact with Gitar



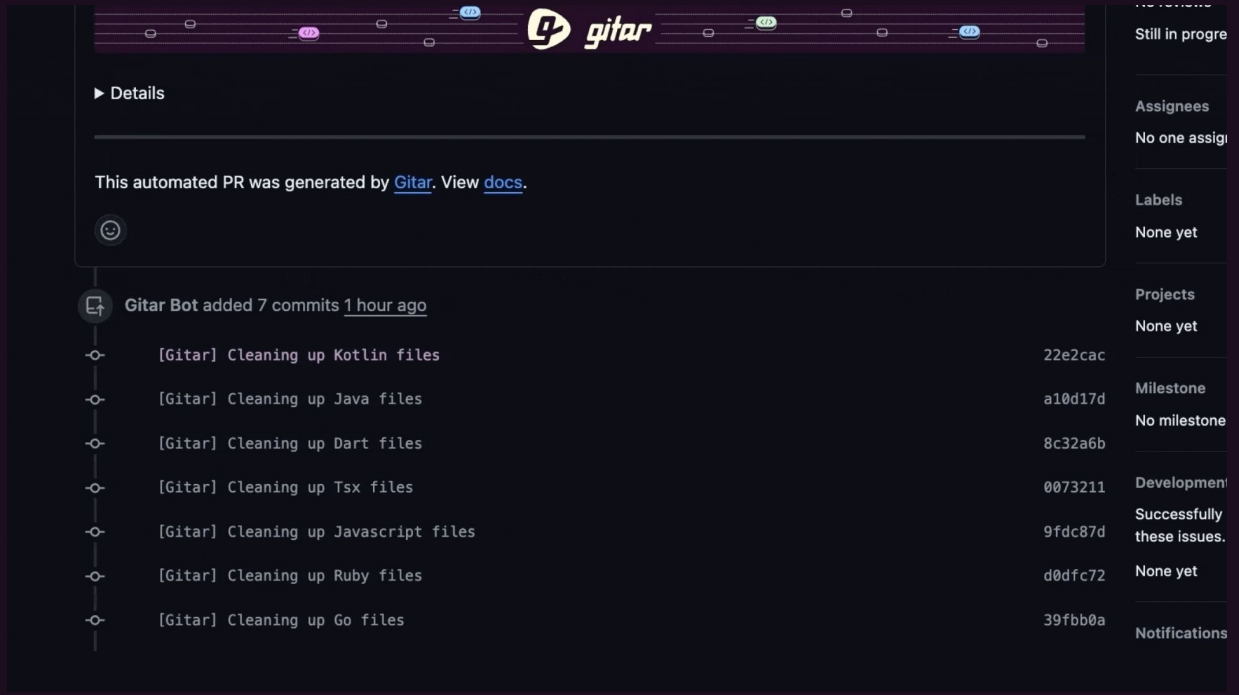
IDE support


SampleJava.java 5
/*
* Copyright (c) 2023-2024 Gitar, Inc.
*/
package co.gitar;
import com.launchdarkly.sdk.*;
import com.launchdarkly.sdk.server.*;
class Sample {
public void sampleMethod1() {
LDClient client = new LDClient("sdk-key-123abc");
LDContext context = LDContext
.builder("context-key-123abc")
.name("Sandy")
.build();
boolean flagValue = client.boolVariation("flag-key-123abc", context, false);
Click Quick Fix for Stale F
if (flagValue) {
// Application code to show the feature
System.out.println("flag-key-123abc is enabled");
} else {
// The code to run if the feature is off
System.out.println("flag-key-123abc is disabled");
}
}
public void sampleMethod2(LDClient client) {
LDContext context = LDContext
.builder("context-key-123abc")

Modern codebases are polyglot

Modern code bases are multi-platform and use many languages + frameworks

Gitar works across languages, frameworks, and large scale repos



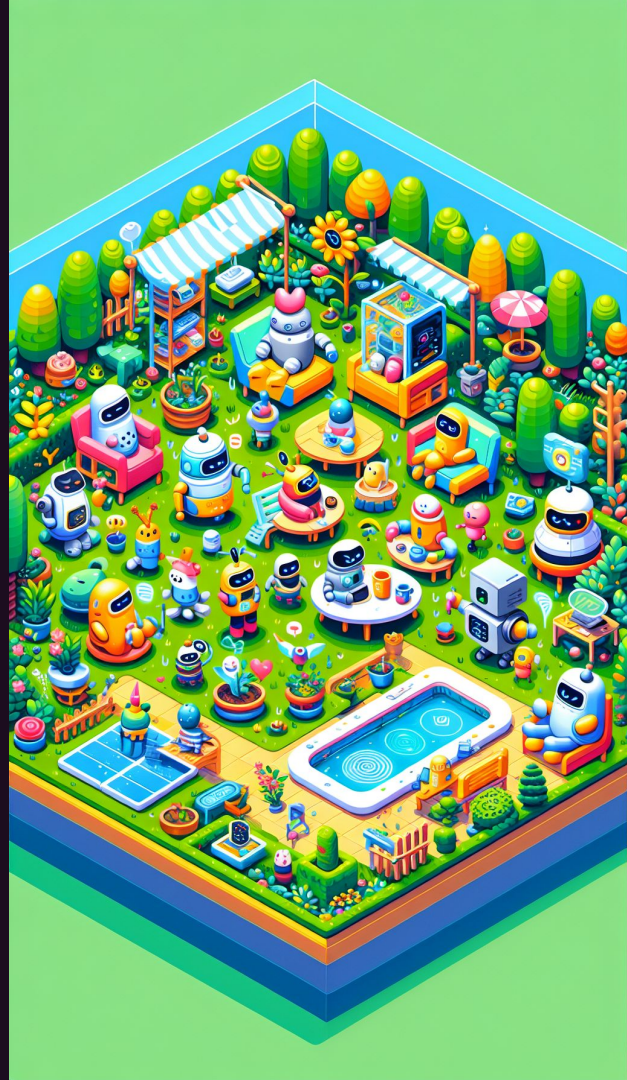
Rich ecosystem integration

Intelligent Platform

Explainable AI

Framework Intelligence

Compiler Foundations



Key Takeaways

- Code maintenance takes too much time
- Automation is key ... but is hard
- Gitar automates code maintenance

gitar.co