Engineering for Software

Amplifying Creativity

Dave Farley

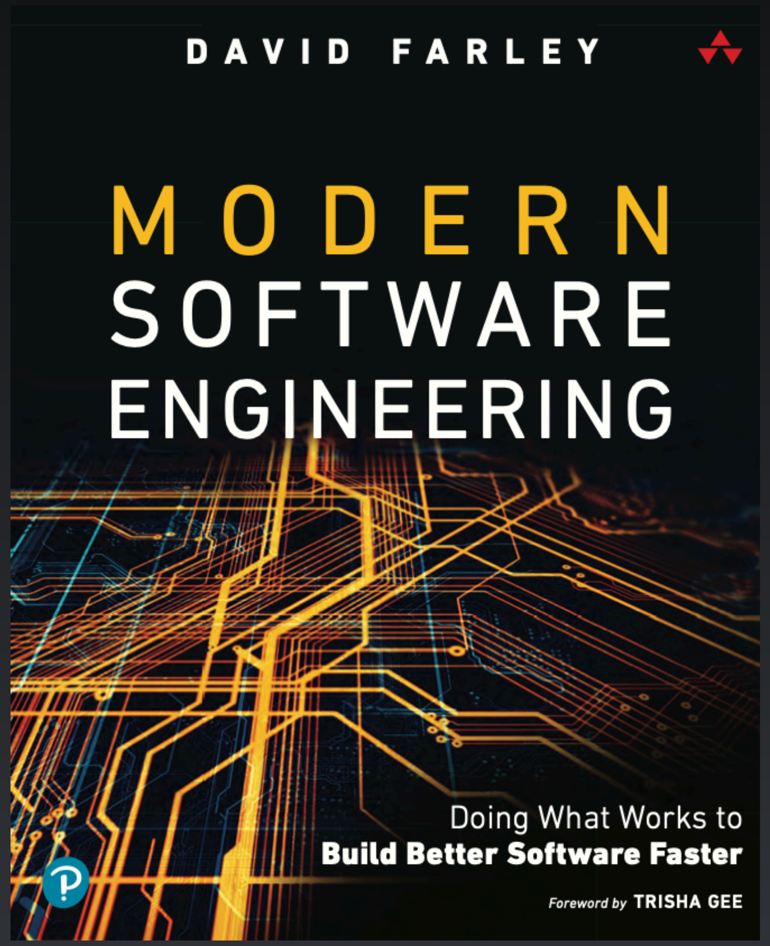
https://www.davefarley.net

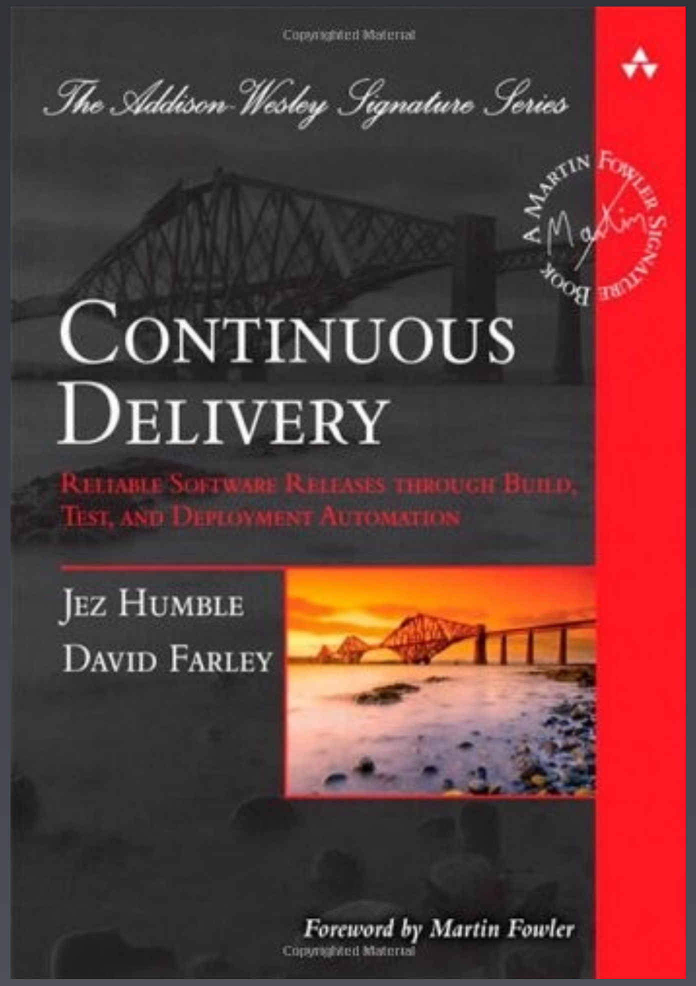
@davefarley77

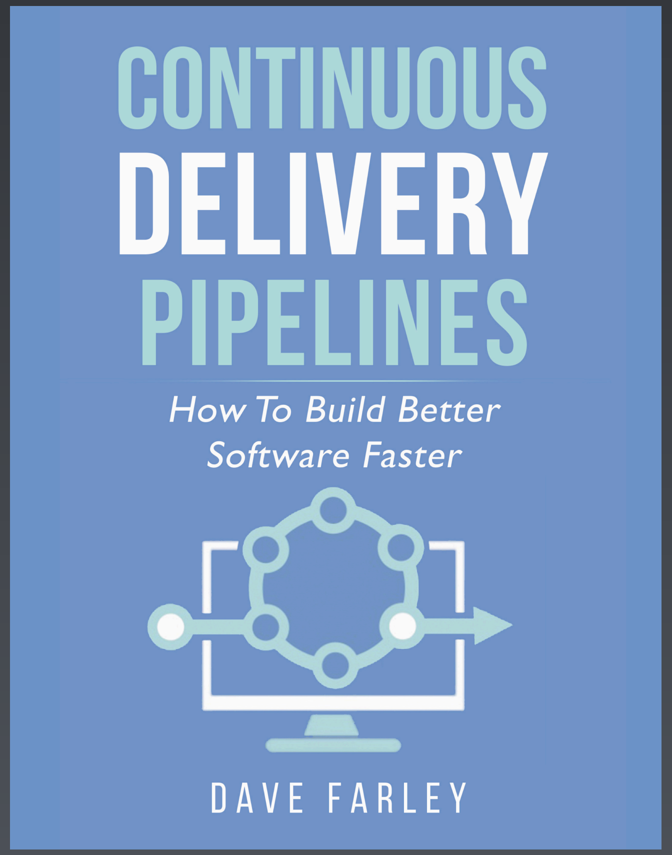
https://bit.ly/CDonYT

Continuous Delivery ltd

http://www.continuous-delivery.co.uk


DAVID FARLEY
MODERN SOFTWARE ENGINEERING
Doing What Works to Build Better Software Faster
Foreword by TRISHA GEE


The Addison-Wesley Signature Series
CONTINUOUS DELIVERY
Reliable Software Releases through Build, Test, and Deployment Automation
Jez Humble
David Farley
Foreword by Martin Fowler


CONTINUOUS DELIVERY PIPELINES
How To Build Better Software Faster
DAVE FARLEY

What Does "Software Engineering" Mean?

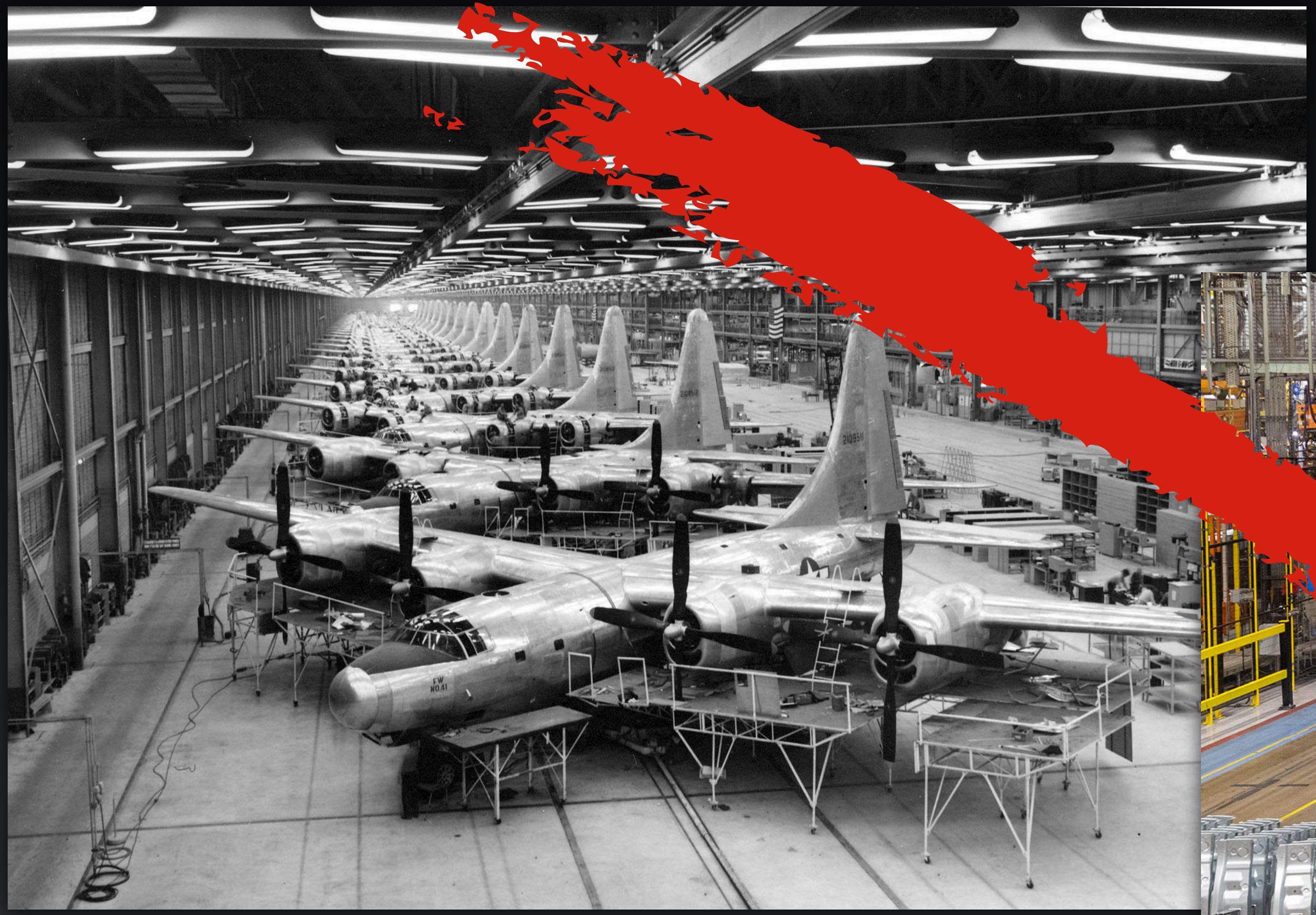
Continuous Delivery ltd

“The Things We Can’t Afford to Get Wrong”

“The Things We Can’t Afford to Get Wrong”

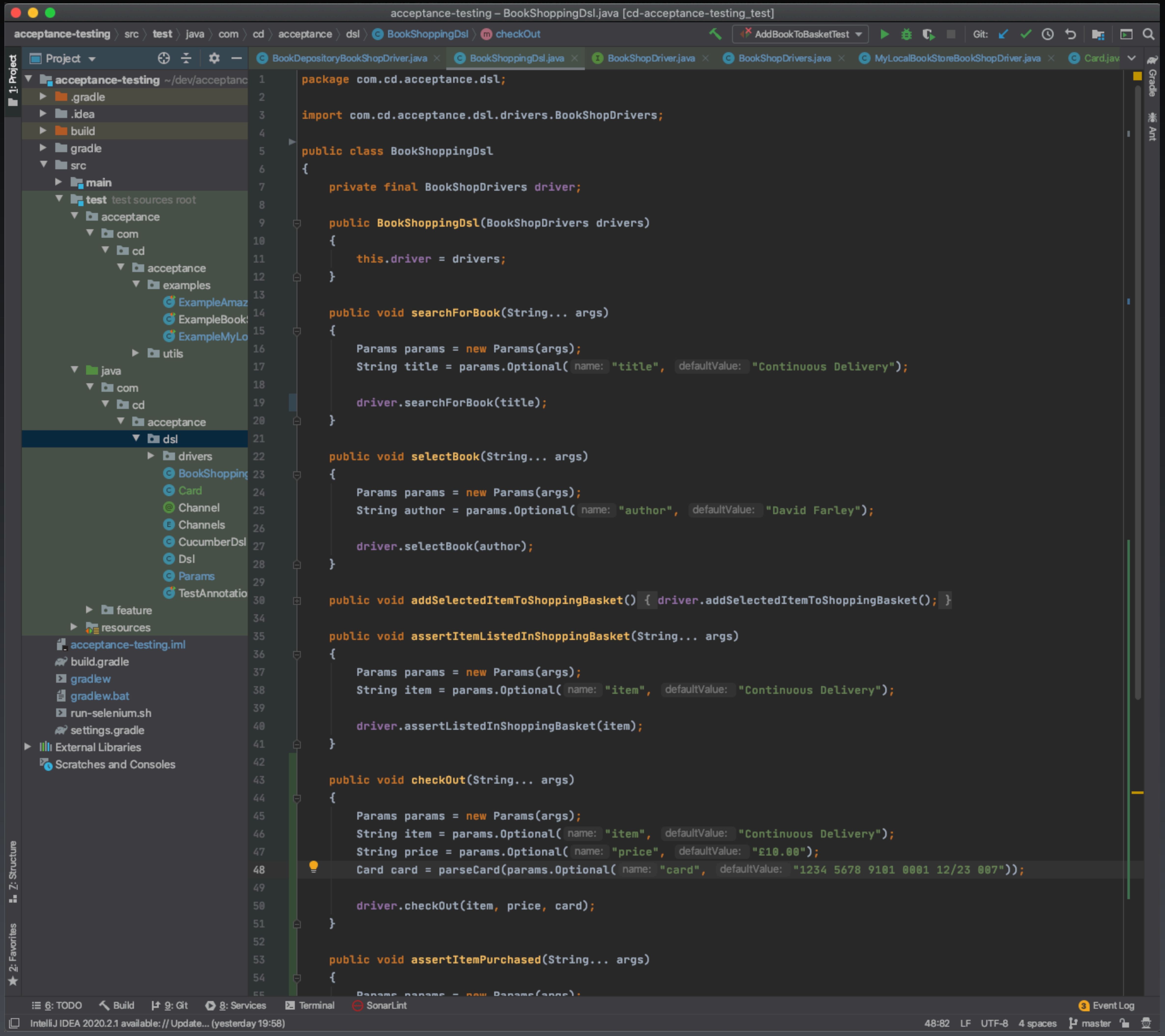
Continuous Delivery ltd

~~“The Things We Can’t Afford to Get Wrong”~~



This is NOT Our Problem!

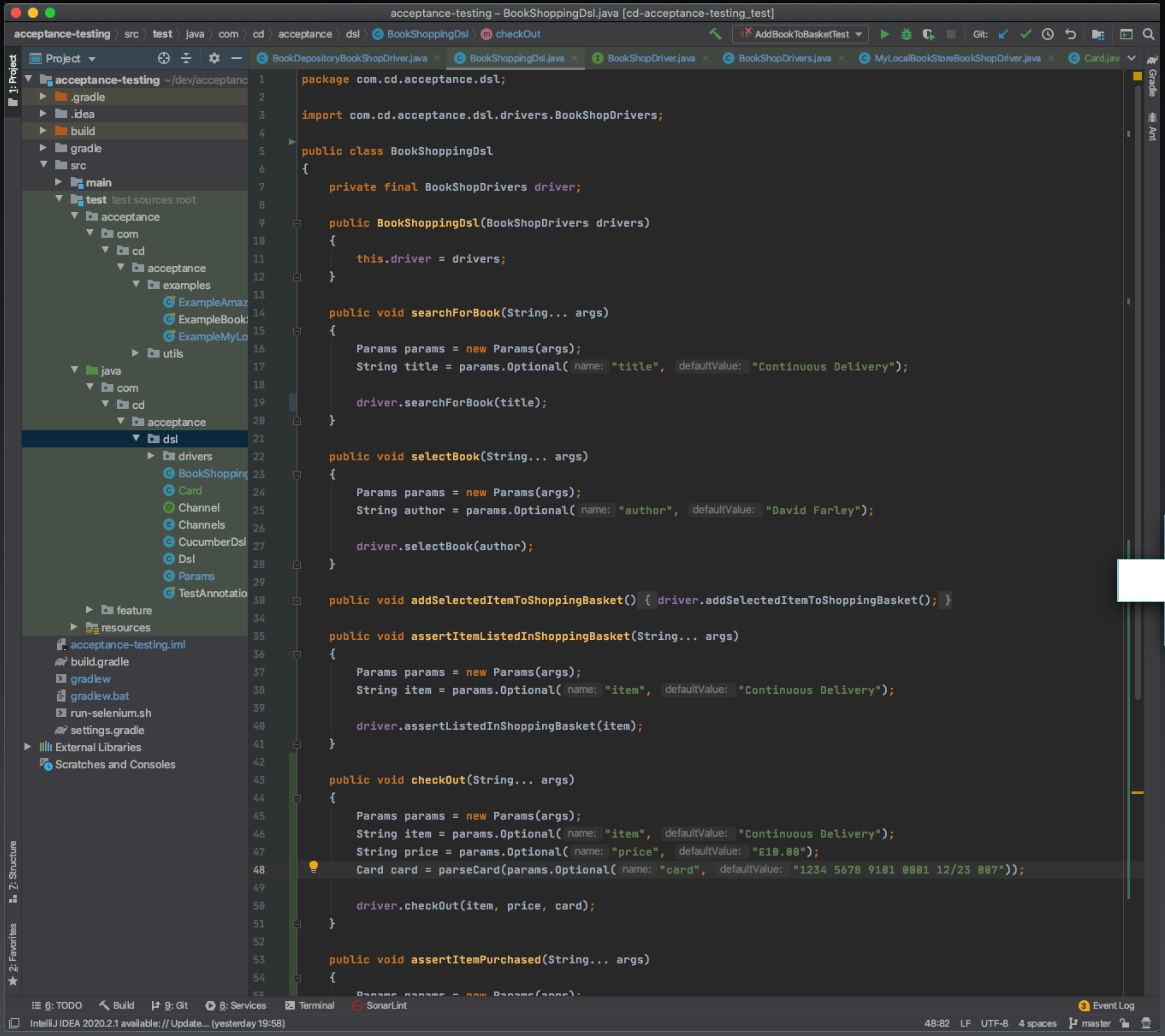
Continuous Delivery ltd



acceptance-testing – BookShoppingDsl.java [cd-acceptance-testing_test]

```java
package com.cd.acceptance.dsl;

import com.cd.acceptance.dsl.drivers.BookShopDrivers;

public class BookShoppingDsl
{
    private final BookShopDrivers driver;

    public BookShoppingDsl(BookShopDrivers drivers)
    {
        this.driver = drivers;
    }

    public void searchForBook(String... args)
    {
        Params params = new Params(args);
        String title = params.Optional( name: "title",  defaultValue: "Continuous Delivery");

        driver.searchForBook(title);
    }

    public void selectBook(String... args)
    {
        Params params = new Params(args);
        String author = params.Optional( name: "author",  defaultValue: "David Farley");

        driver.selectBook(author);
    }

    public void addSelectedItemToShoppingBasket() { driver.addSelectedItemToShoppingBasket(); }

    public void assertItemListedInShoppingBasket(String... args)
    {
        Params params = new Params(args);
        String item = params.Optional( name: "item",  defaultValue: "Continuous Delivery");

        driver.assertListedInShoppingBasket(item);
    }

    public void checkOut(String... args)
    {
        Params params = new Params(args);
        String item = params.Optional( name: "item",  defaultValue: "Continuous Delivery");
        String price = params.Optional( name: "price",  defaultValue: "£10.00");
        Card card = parseCard(params.Optional( name: "card",  defaultValue: "1234 5678 9101 0001 12/23 007"));

        driver.checkOut(item, price, card);
    }

    public void assertItemPurchased(String... args)
    {
        Params params = new Params(args);
```

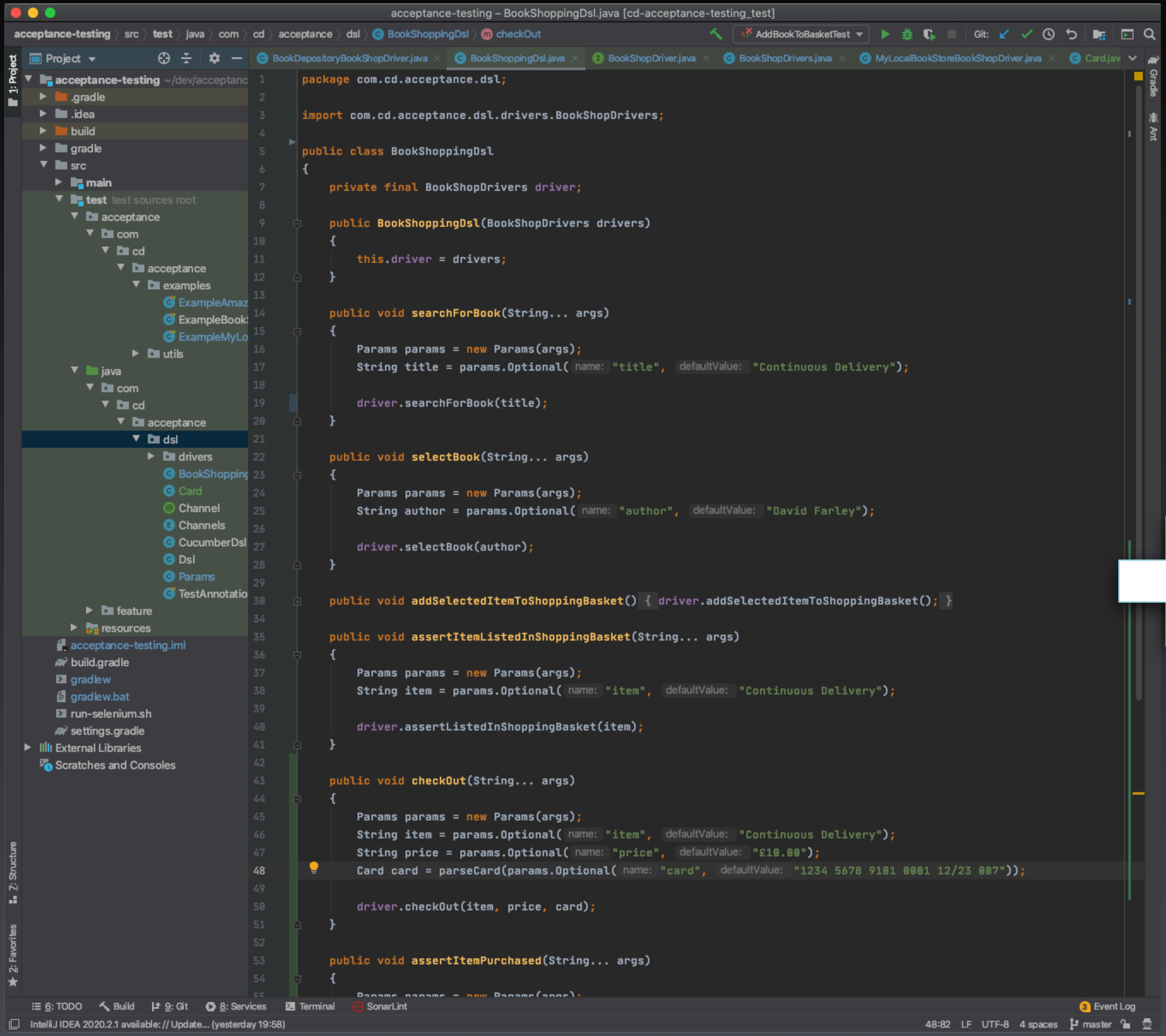
Continuous
Delivery ltd



acceptance-testing – BookShoppingDsl.java [cd-acceptance-testing_test]

```java
package com.cd.acceptance.dsl;

import com.cd.acceptance.dsl.drivers.BookShopDrivers;

public class BookShoppingDsl
{
    private final BookShopDrivers driver;

    public BookShoppingDsl(BookShopDrivers drivers)
    {
        this.driver = drivers;
    }

    public void searchForBook(String... args)
    {
        Params params = new Params(args);
        String title = params.Optional("title", "Continuous Delivery");

        driver.searchForBook(title);
    }

    public void selectBook(String... args)
    {
        Params params = new Params(args);
        String author = params.Optional("author", "David Farley");

        driver.selectBook(author);
    }

    public void addSelectedItemToShoppingBasket() { driver.addSelectedItemToShoppingBasket(); }

    public void assertItemListedInShoppingBasket(String... args)
    {
        Params params = new Params(args);
        String item = params.Optional("item", "Continuous Delivery");

        driver.assertListedInShoppingBasket(item);
    }

    public void checkOut(String... args)
    {
        Params params = new Params(args);
        String item = params.Optional("item", "Continuous Delivery");
        String price = params.Optional("price", "£10.00");
        Card card = parseCard(params.Optional("card", "1234 5678 9101 0001 12/23 007"));

        driver.checkOut(item, price, card);
    }

    public void assertItemPurchased(String... args)
    {
```

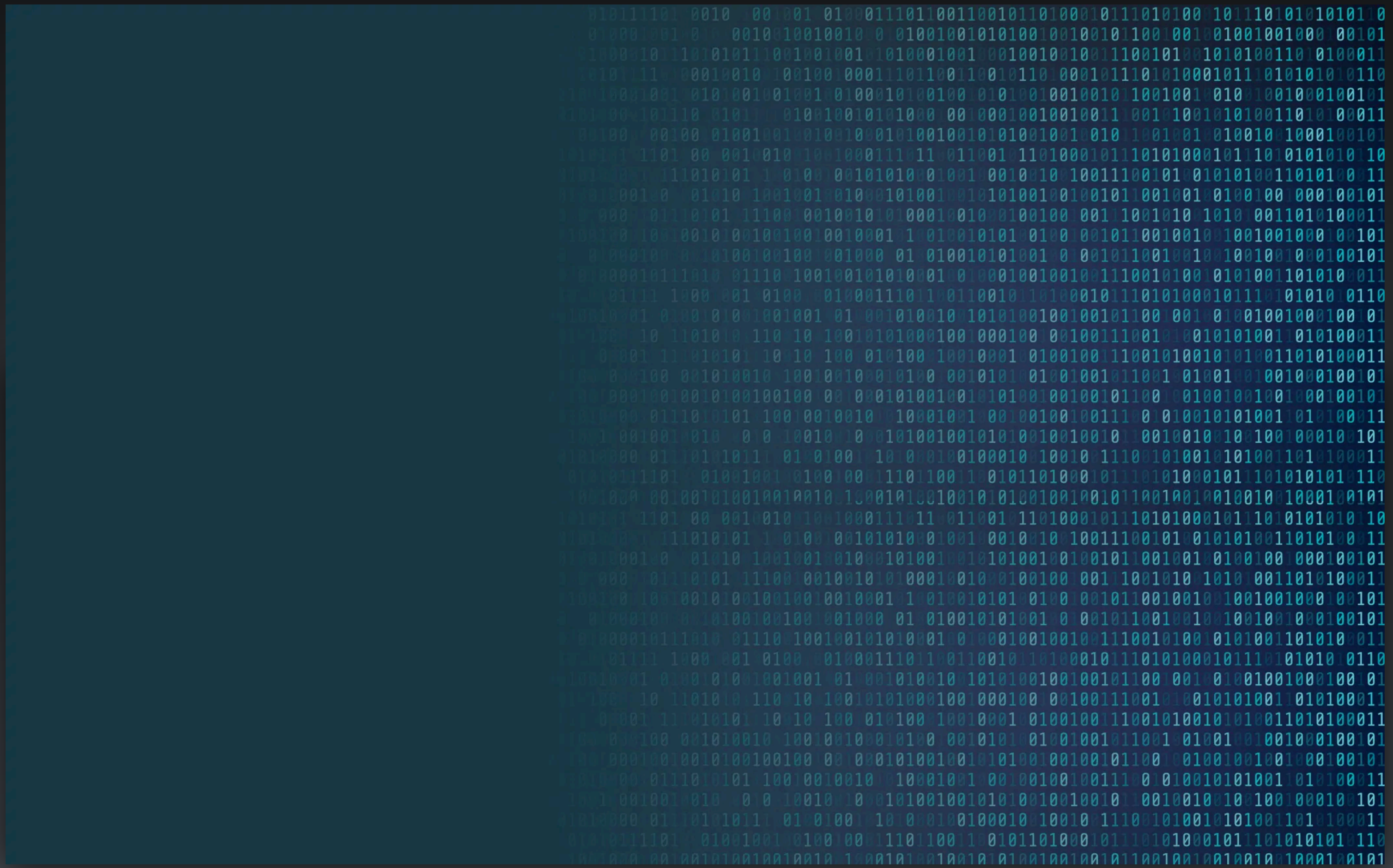
Continuous Delivery ltd

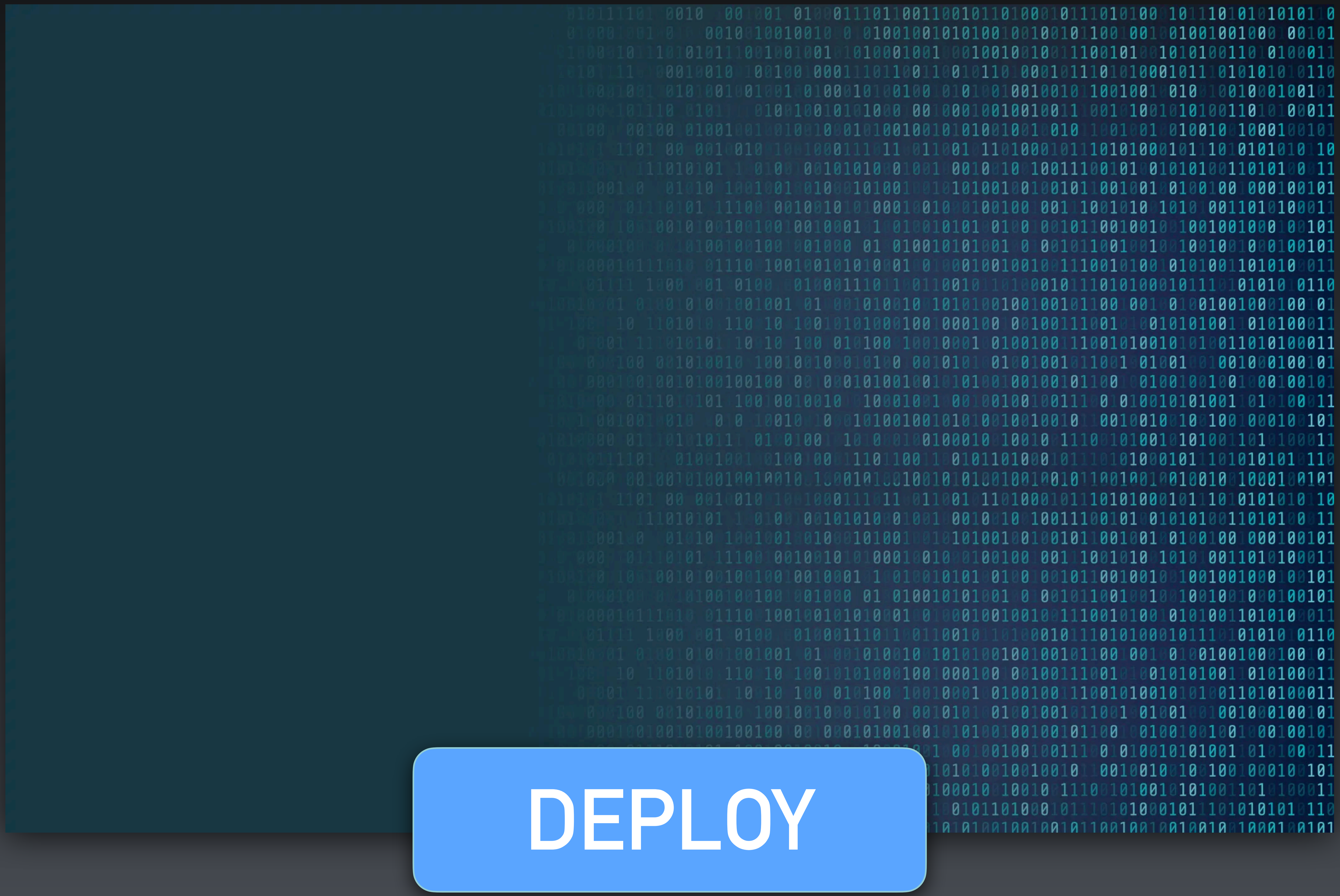

acceptance-testing – BookShoppingDsl.java [cd-acceptance-testing_test]
BookDepositoryBookShopDriver.java
BookShoppingDsl.java
BookShopDriver.java
BookShopDrivers.java
MyLocalBookStoreBookShopDriver.java
AddBookToBasketTest

```java
package com.cd.acceptance.dsl;

import com.cd.acceptance.dsl.drivers.BookShopDrivers;

public class BookShoppingDsl
{
    private final BookShopDrivers driver;

    public BookShoppingDsl(BookShopDrivers drivers)
    {
        this.driver = drivers;
    }

    public void searchForBook(String... args)
    {
        Params params = new Params(args);
        String title = params.Optional("title", "Continuous Delivery");

        driver.searchForBook(title);
    }

    public void selectBook(String... args)
    {
        Params params = new Params(args);
        String author = params.Optional("author", "David Farley");

        driver.selectBook(author);
    }

    public void addSelectedItemToShoppingBasket() { driver.addSelectedItemToShoppingBasket(); }

    public void assertItemListedInShoppingBasket(String... args)
    {
        Params params = new Params(args);
        String item = params.Optional("item", "Continuous Delivery");

        driver.assertListedInShoppingBasket(item);
    }

    public void checkOut(String... args)
    {
        Params params = new Params(args);
        String item = params.Optional("item", "Continuous Delivery");
        String price = params.Optional("price", "£10.00");
        Card card = parseCard(params.Optional("card", "1234 5678 9101 0001 12/23 007"));

        driver.checkOut(item, price, card);
    }

    public void assertItemPurchased(String... args)
    {
```

Continuous Delivery ltd

Continuous Delivery ltd




DEPLOY

We Can Recreate Entire Systems for Free!

Continuous Delivery ltd

Engineering:

"Designing, Building & Repairing things in a Principled way"

Alan Kay

Continuous Delivery ltd

Engineering: Design for Failure

Alan Kay

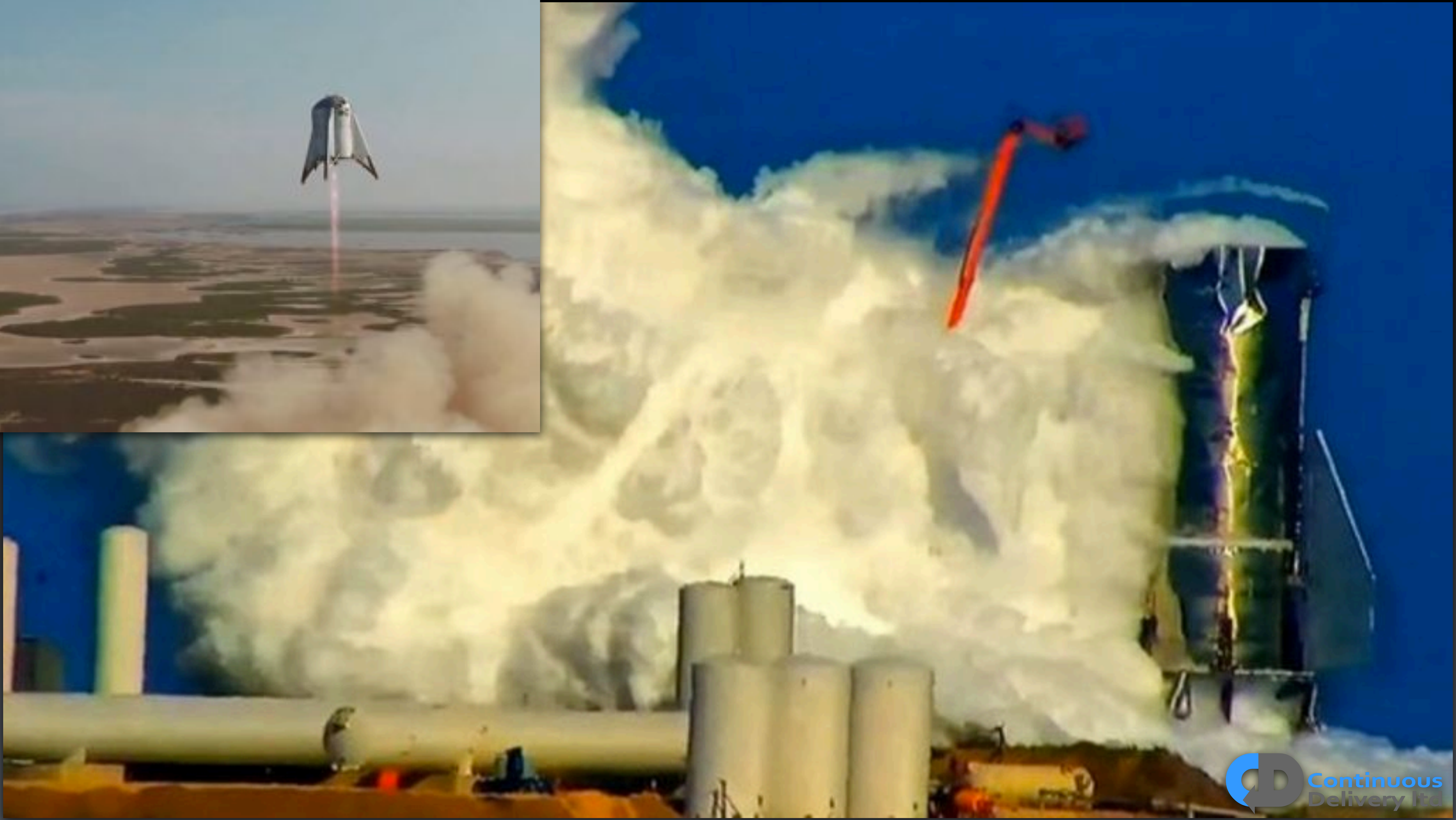
Continuous Delivery ltd

All Engineering is not the same!

Continuous Delivery ltd

Continuous Delivery ltd

Continuous Delivery ltd.



Continuous Delivery ltd

Engineering *IS* About Exploration & Discovery

Continuous Delivery ltd

Engineering *IS* About Exploration & Discovery

Continuous Delivery ltd

Software Development *IS* About Exploration & Discovery

Continuous Delivery ltd

Optimise for Learning

Continuous Delivery ltd

Optimise for Learning

Iteration

Continuous Delivery ltd

Optimise for Learning

Iteration

Feedback

Continuous Delivery ltd

Optimise for Learning

Iteration

Feedback

Incremental

Continuous Delivery ltd

Optimise for Learning

Iteration

Feedback

Incremental

Experimental

Continuous Delivery ltd

Optimise for Learning

Iteration

Feedback

Incremental

Experimental

Empirical

Software Development is also about Managing Complexity

Continuous Delivery ltd

Continuous Delivery ltd

Optimise to Manage Complexity

Continuous Delivery ltd

Optimise to Manage Complexity

Modularity

Optimise to Manage Complexity

Modularity

Abstraction

Optimise to Manage Complexity

Modularity

Abstraction

Separation of Concerns

Continuous Delivery ltd

Optimise to Manage Complexity

Modularity
Abstraction
Separation of Concerns
Loose Coupling

Continuous Delivery ltd

Optimise to Manage Complexity

Modularity

Abstraction

Separation of Concerns

Loose Coupling

Cohesion

Continuous Delivery ltd

Principles of Applying Engineering Thinking

- Optimise for Learning
- Optimise to Manage/Limit Complexity
- Control the Variables
- Make Evidence Based Decisions
  (Run the Experiments)
- Never Assume You Have the Correct Answer
- Find Ways to Falsify Ideas Simply
  (More Experiments!)

Continuous Delivery ltd

Testability

Continuous Delivery ltd

What Do We Value In Good Code?

Continuous Delivery ltd

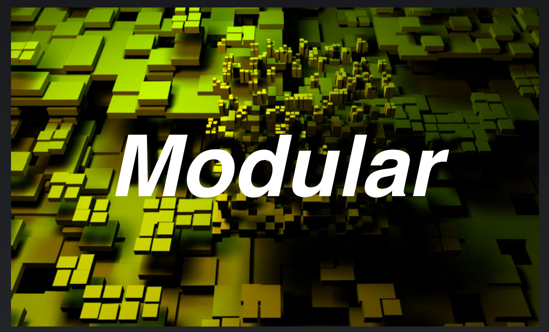
What Do We Value In Good Code?

- It Has to Work!

Continuous Delivery ltd

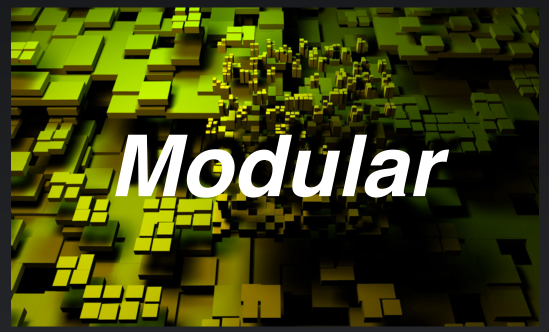
What Do We Value In Good Code?

- It Has to Work!
- Modular



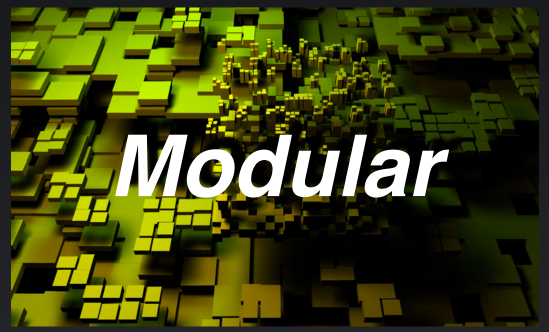
What Do We Value In Good Code?

- It Has to Work!
- Modular
- Loosely-coupled



What Do We Value In Good Code?

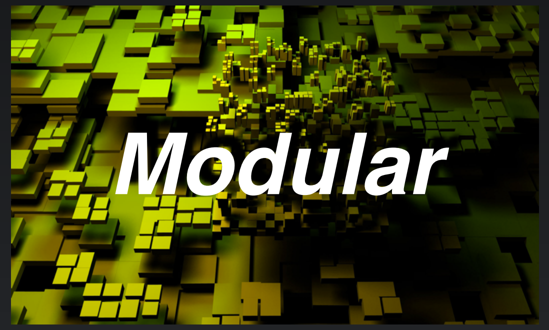
- It Has to Work!
- Modular
- Loosely-coupled
- High-Cohesion

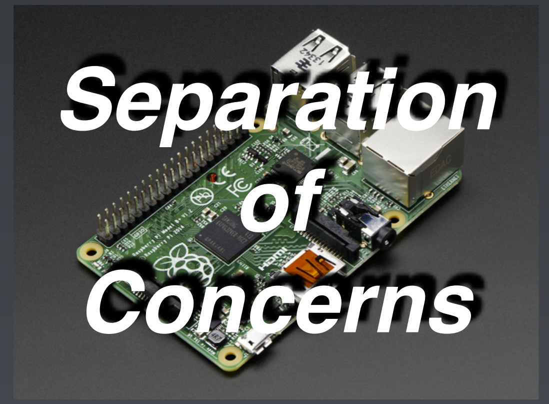


Continuous Delivery ltd

What Do We Value In Good Code?

- It Has to Work!
- Modular
- Loosely-coupled
- High-Cohesion
- Good Separation of Concerns

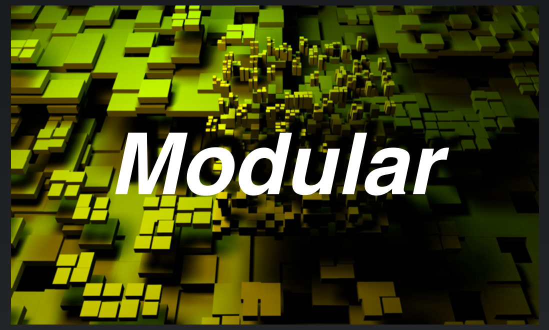


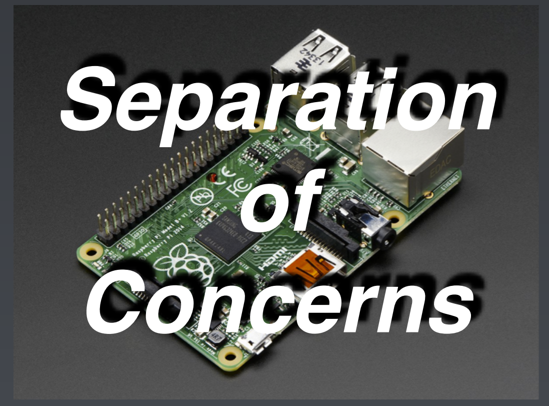


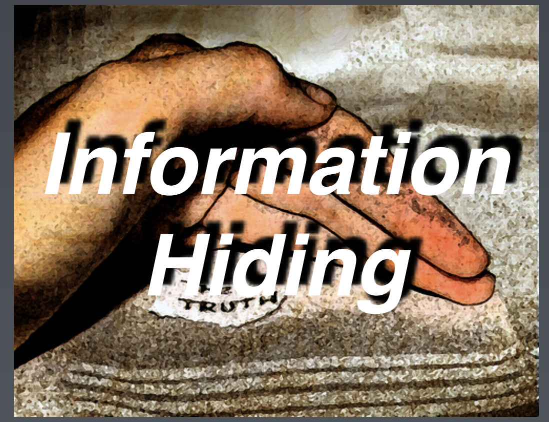
Continuous Delivery ltd

What Do We Value In Good Code?

- It Has to Work!
- Modular
- Loosely-coupled
- High-Cohesion
- Good Separation of Concerns
- Exhibits Information Hiding







Continuous Delivery ltd

What Drives Quality? - Before TDD

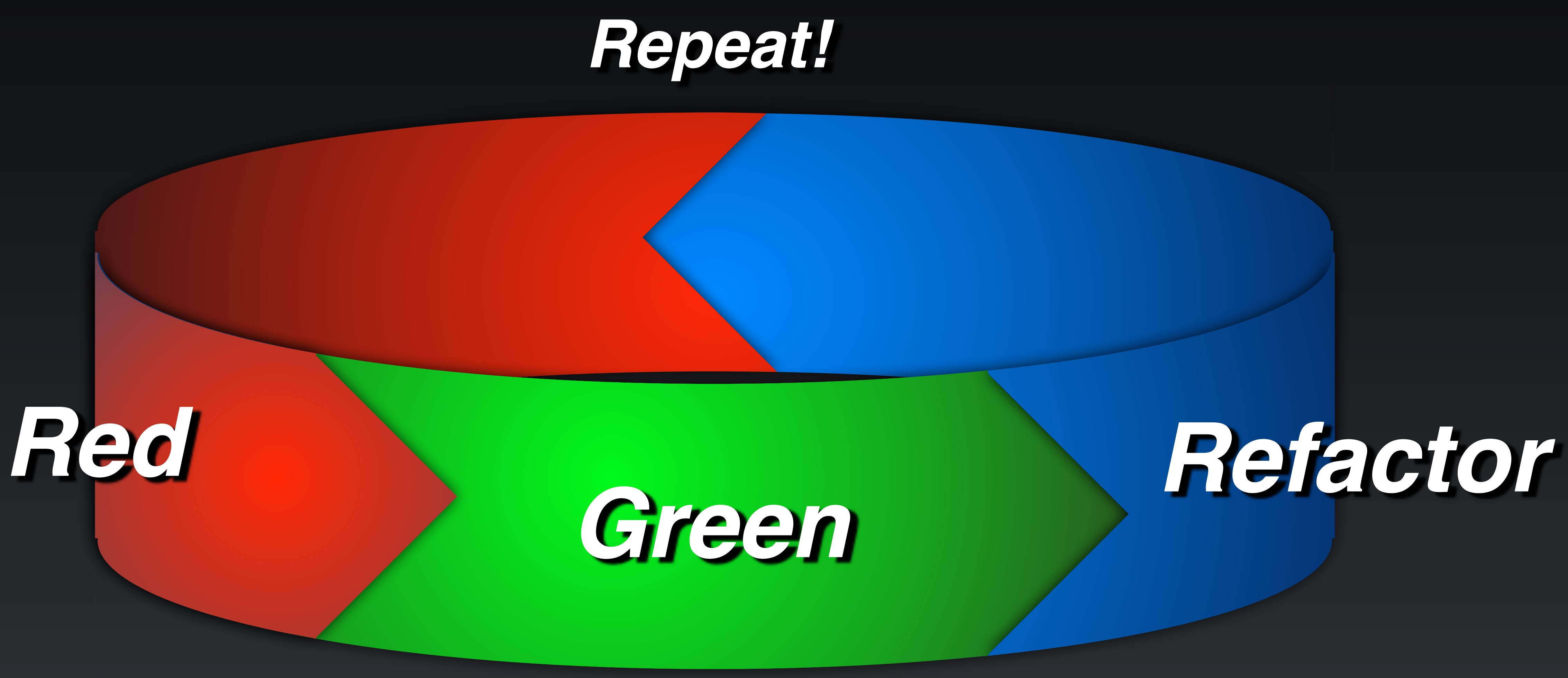
What Drives Quality? - Before TDD

The Skill, Experience and Integrity of an individual programmer.

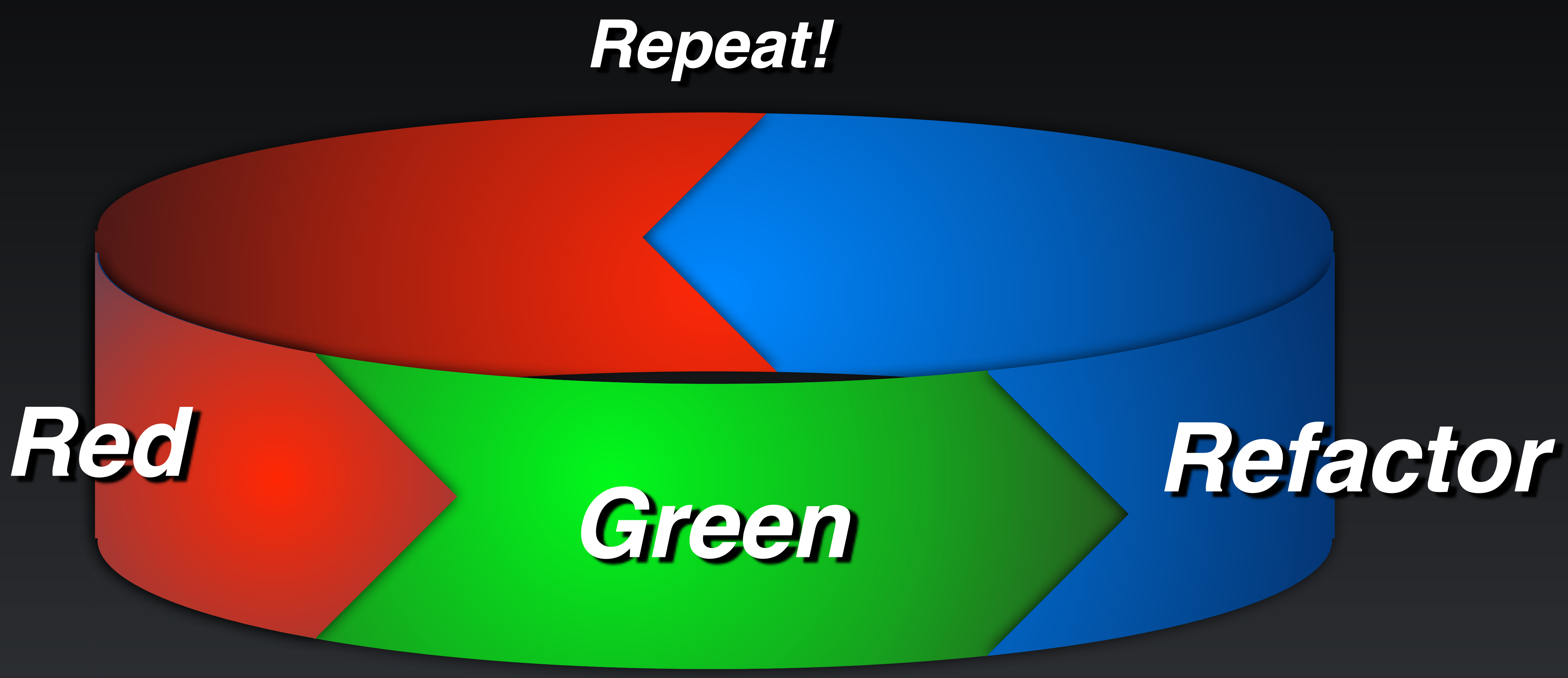
Continuous Delivery ltd

TDD is…

TDD is…

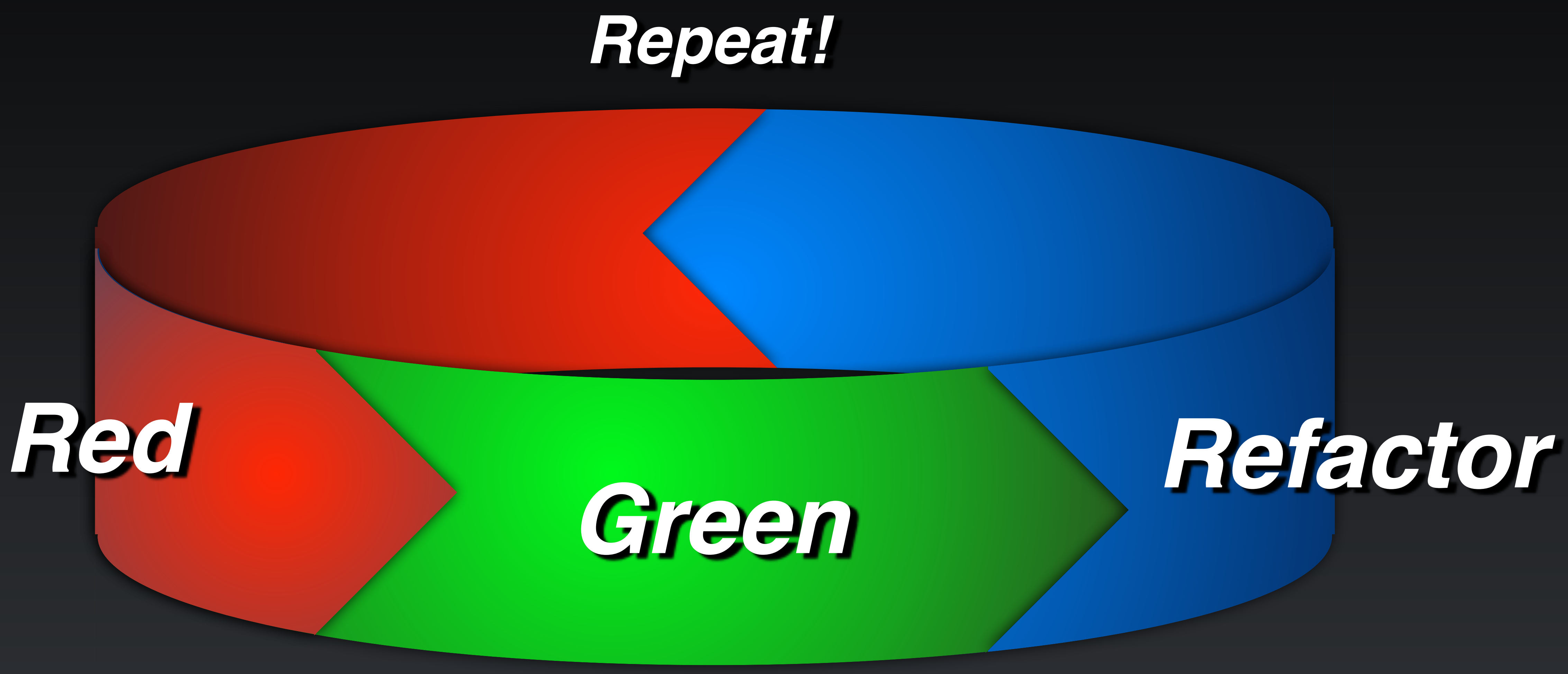


TDD is…



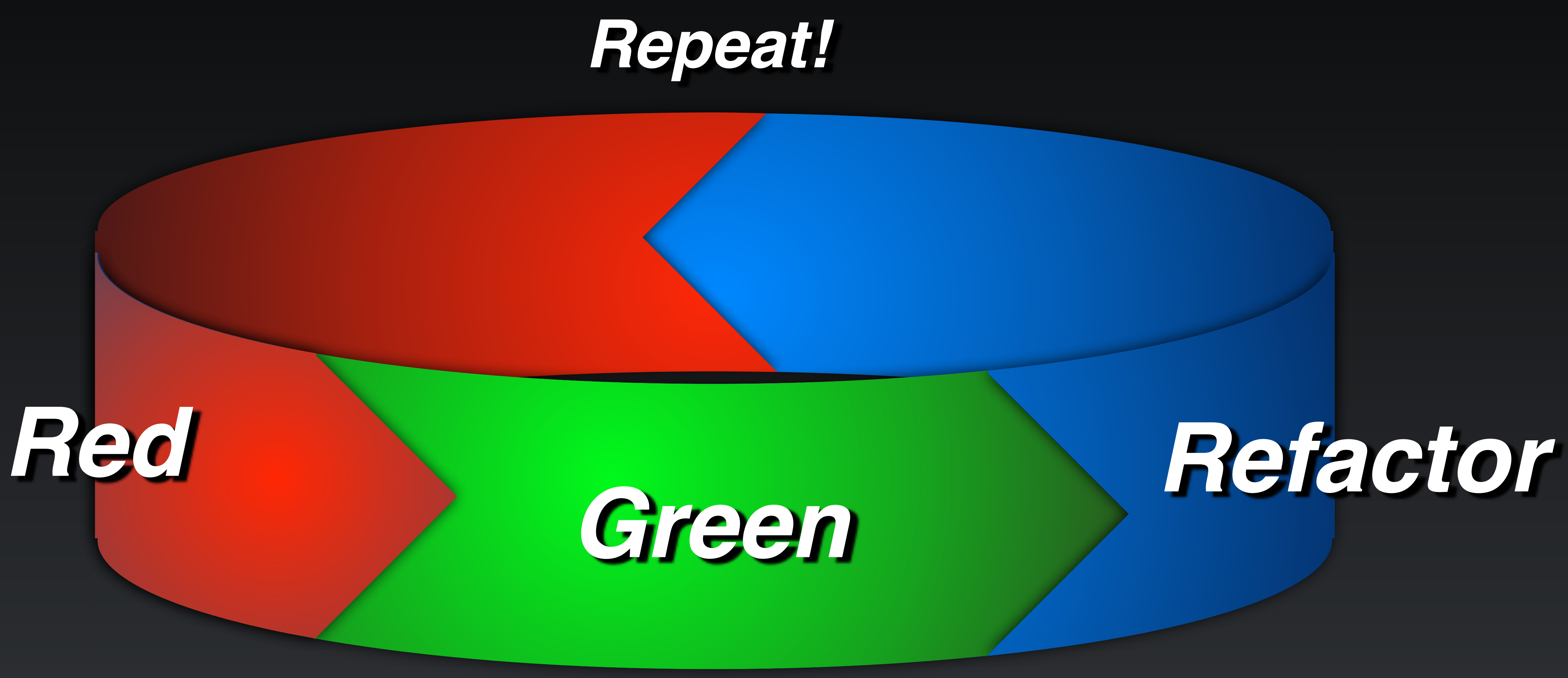
- Write a Test - See it Fail

TDD is…



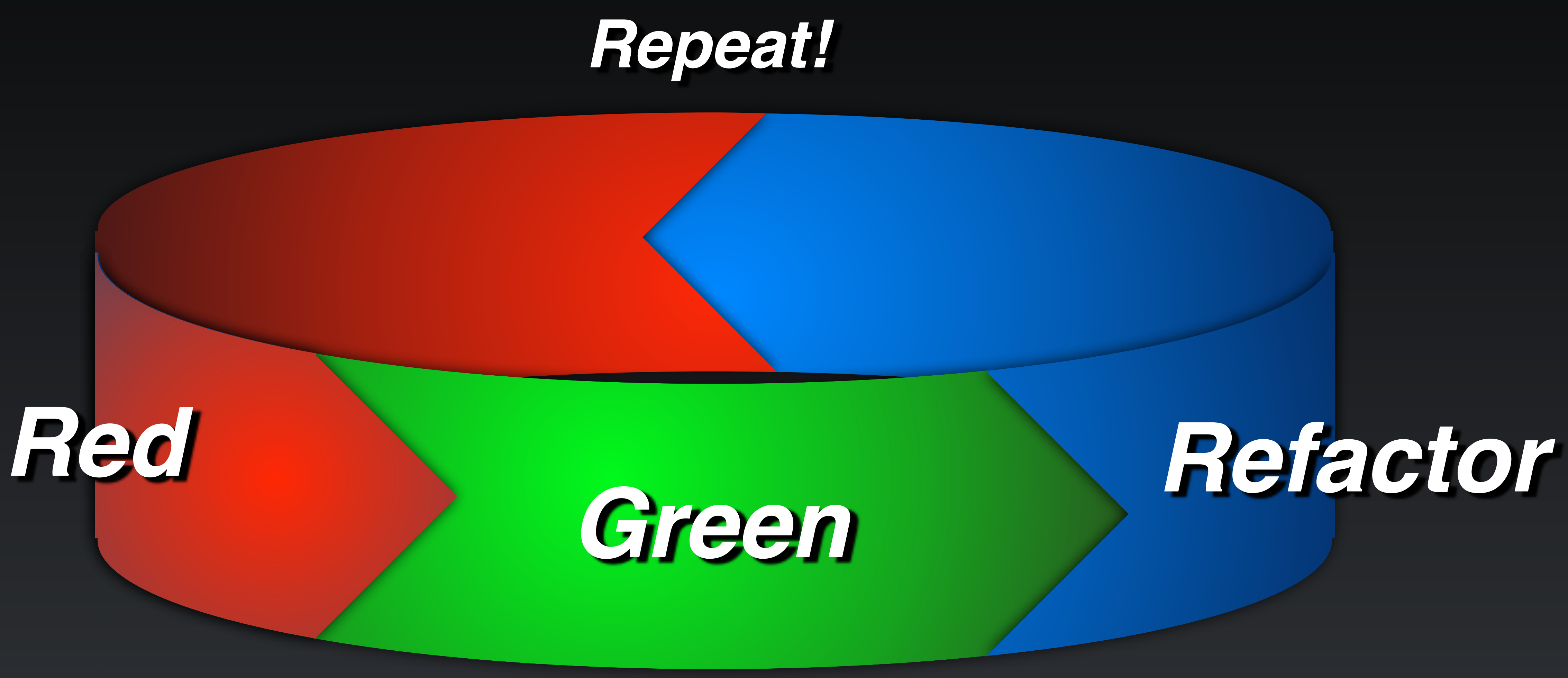
- Write a Test - See it Fail
- Write Code to Make the Test Pass - See it Pass

TDD is…



- Write a Test - See it Fail
- Write Code to Make the Test Pass - See it Pass
- Modify the Code to Make it Clean and Elegant

TDD is…



- Write a Test - See it Fail
- Write Code to Make the Test Pass - See it Pass
- Modify the Code to Make it Clean and Elegant
- Next Test…

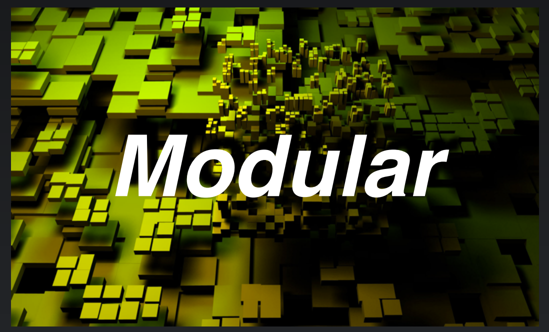
What Makes Code Testable?

What Makes Code Testable?

- It Has to Work!

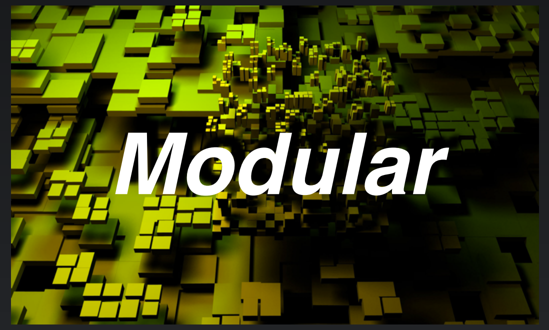
What Makes Code Testable?

- It Has to Work!
- Modular



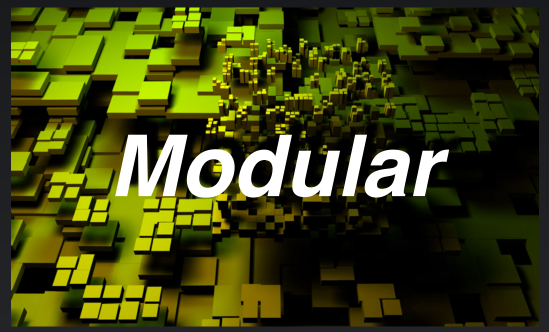
What Makes Code Testable?

- It Has to Work!
- Modular
- Loosely-coupled



What Makes Code Testable?

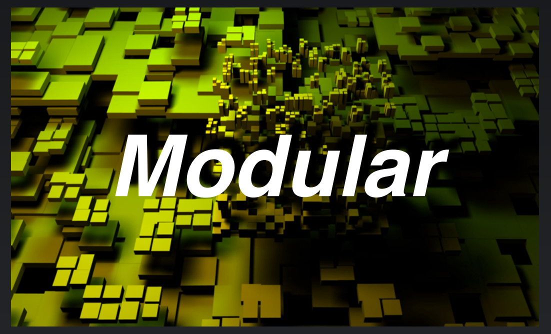
- It Has to Work!
- Modular
- Loosely-coupled
- High-Cohesion

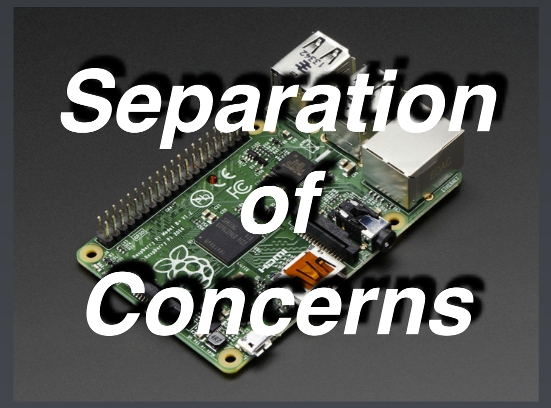


Continuous Delivery ltd

What Makes Code Testable?

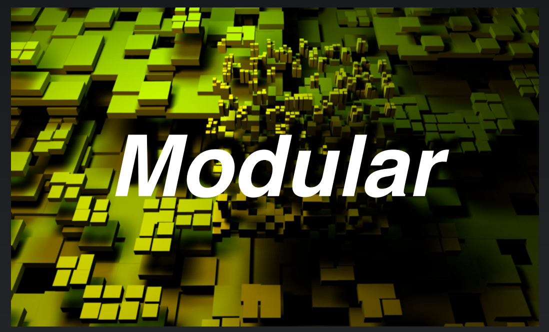
- It Has to Work!
- Modular
- Loosely-coupled
- High-Cohesion
- Good Separation of Concerns

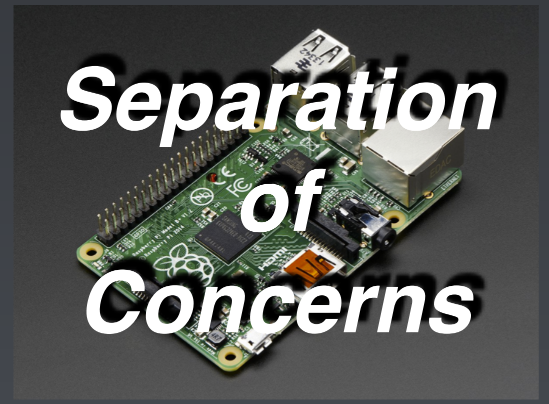


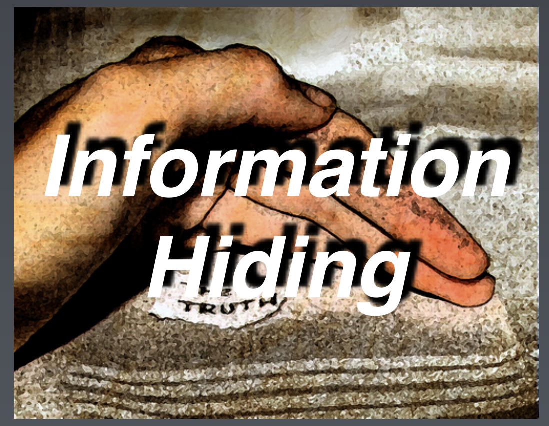


What Makes Code Testable?

- It Has to Work!
- Modular
- Loosely-coupled
- High-Cohesion
- Good Separation of Concerns
- Exhibits Information Hiding







Continuous Delivery ltd

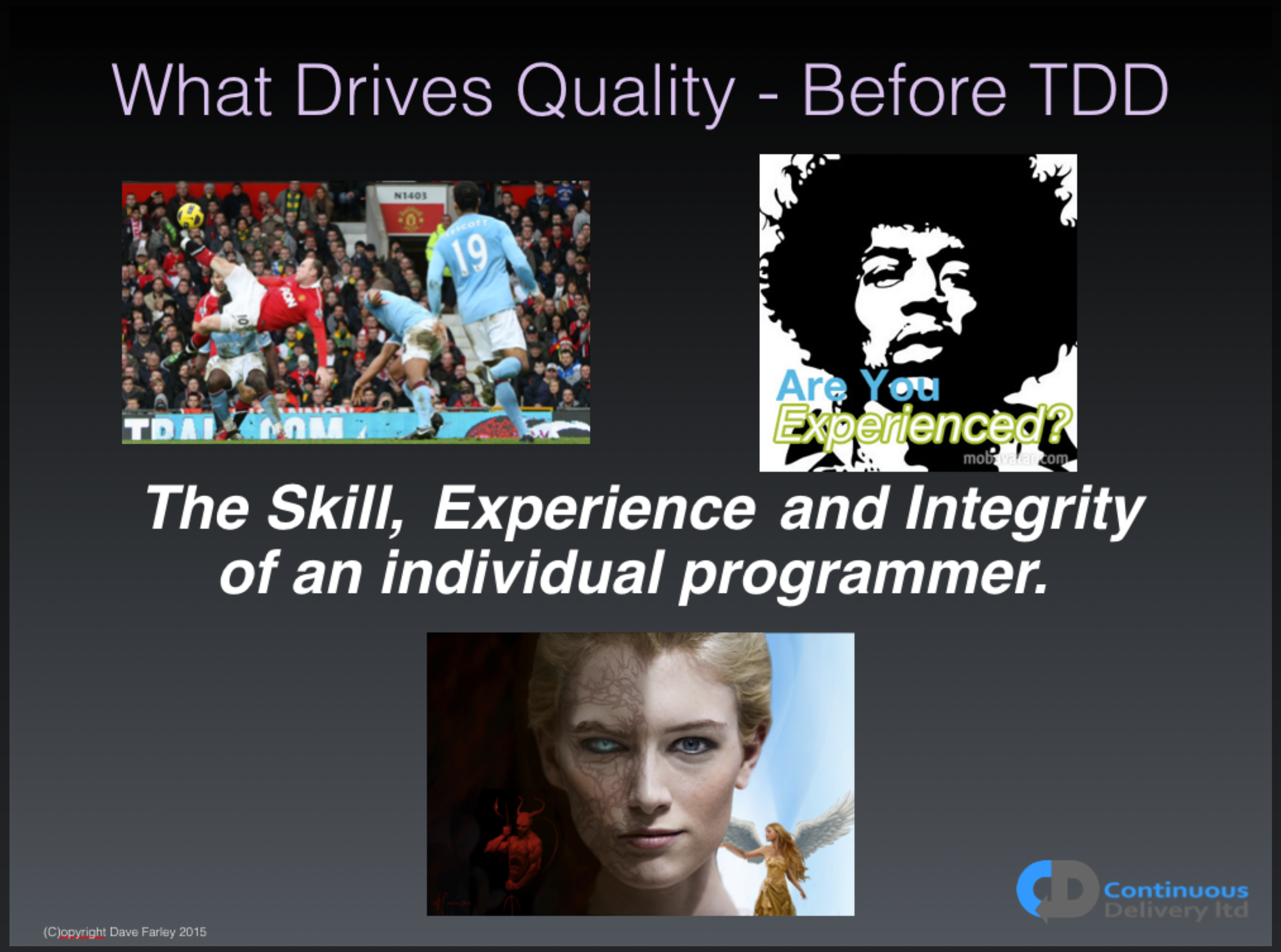
What Drives Quality - Before TDD

The Skill, Experience and Integrity of an individual programmer.

What Drives Quality - Before TDD

Are You
Experienced?

The Skill, Experience and Integrity of an individual programmer.

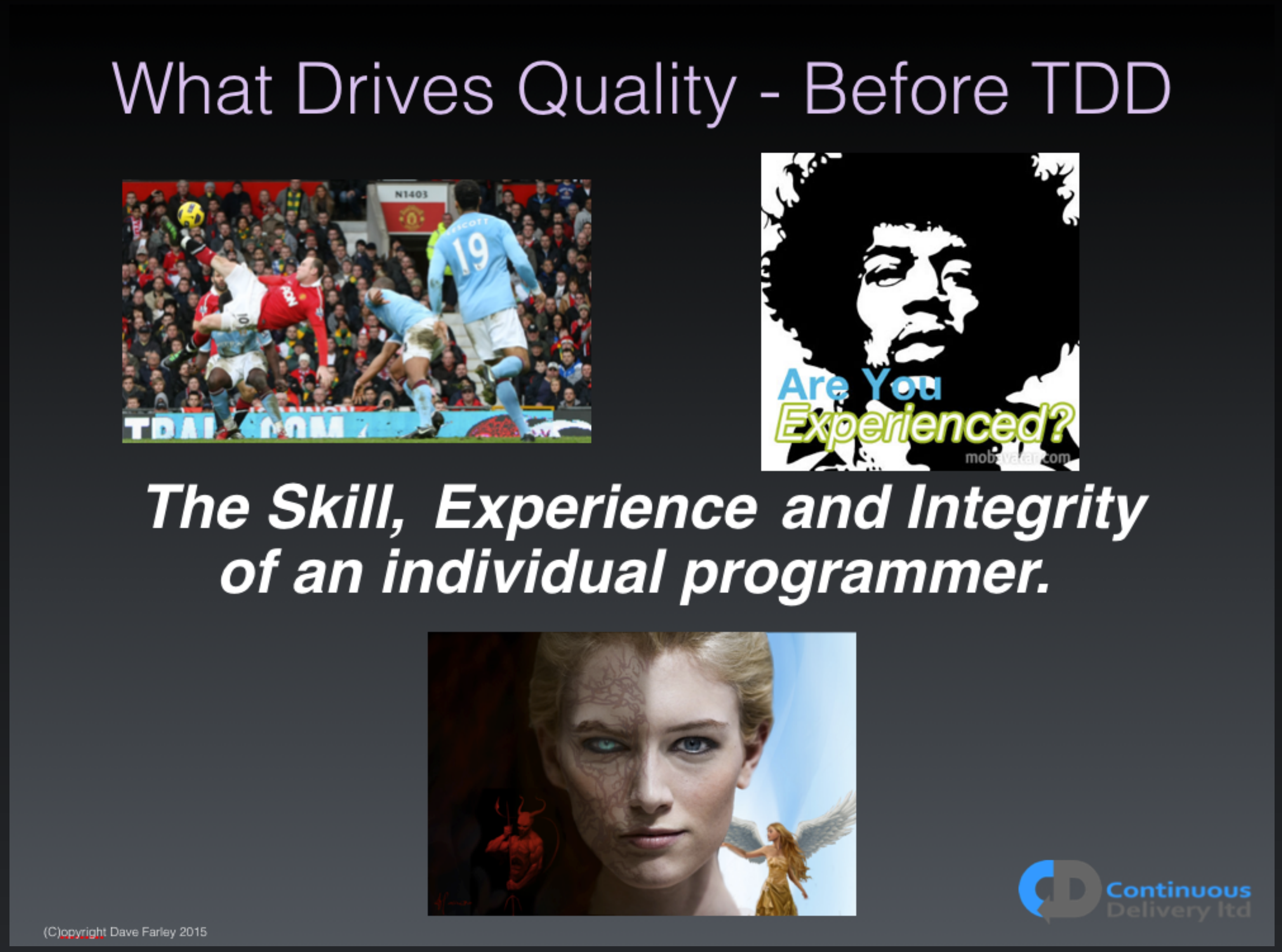


What Drives Quality - Before TDD

Are You
Experienced?

The Skill, Experience and Integrity of an individual programmer.

+

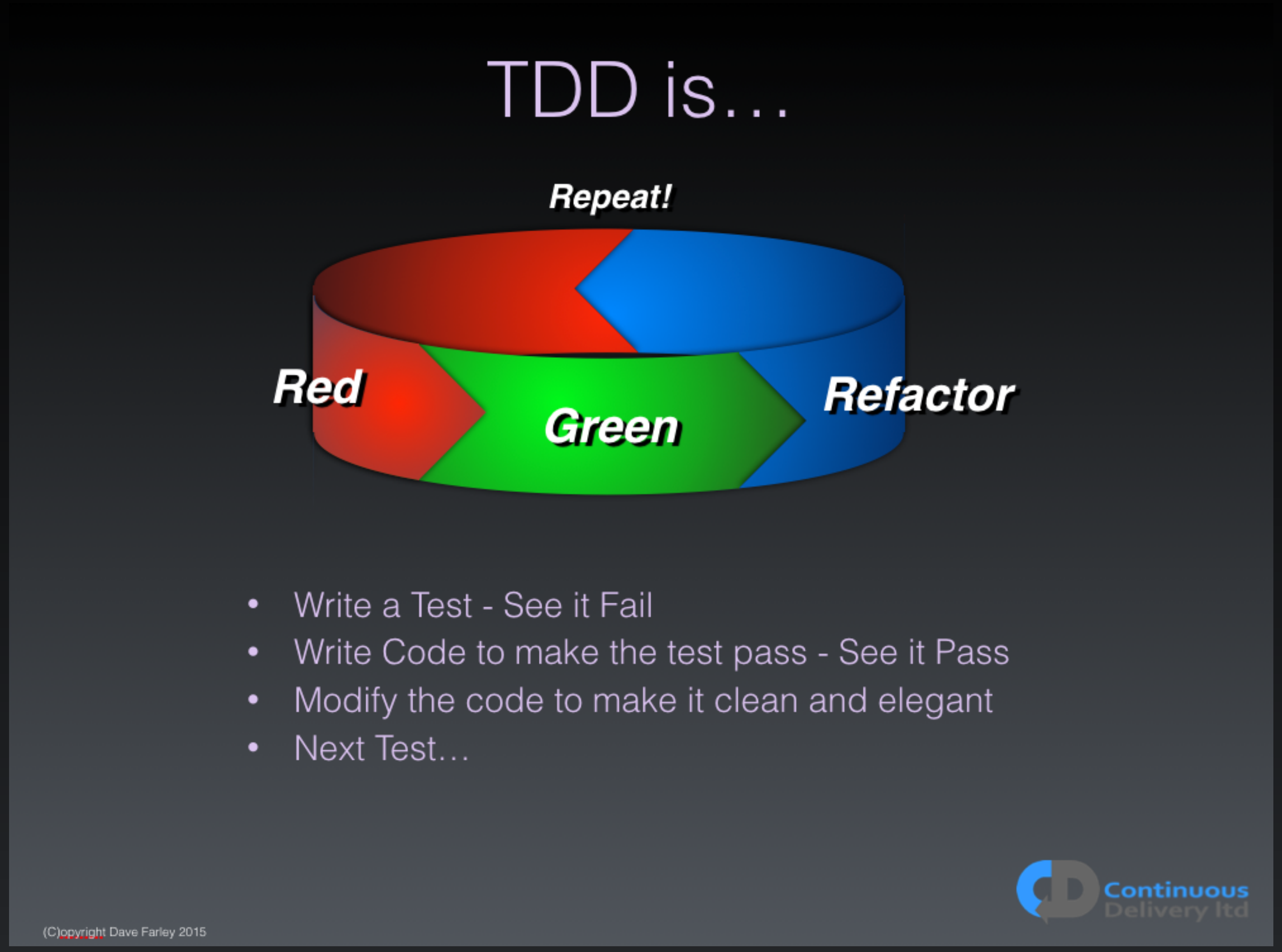


What Drives Quality - Before TDD

Are You Experienced?

The Skill, Experience and Integrity of an individual programmer.

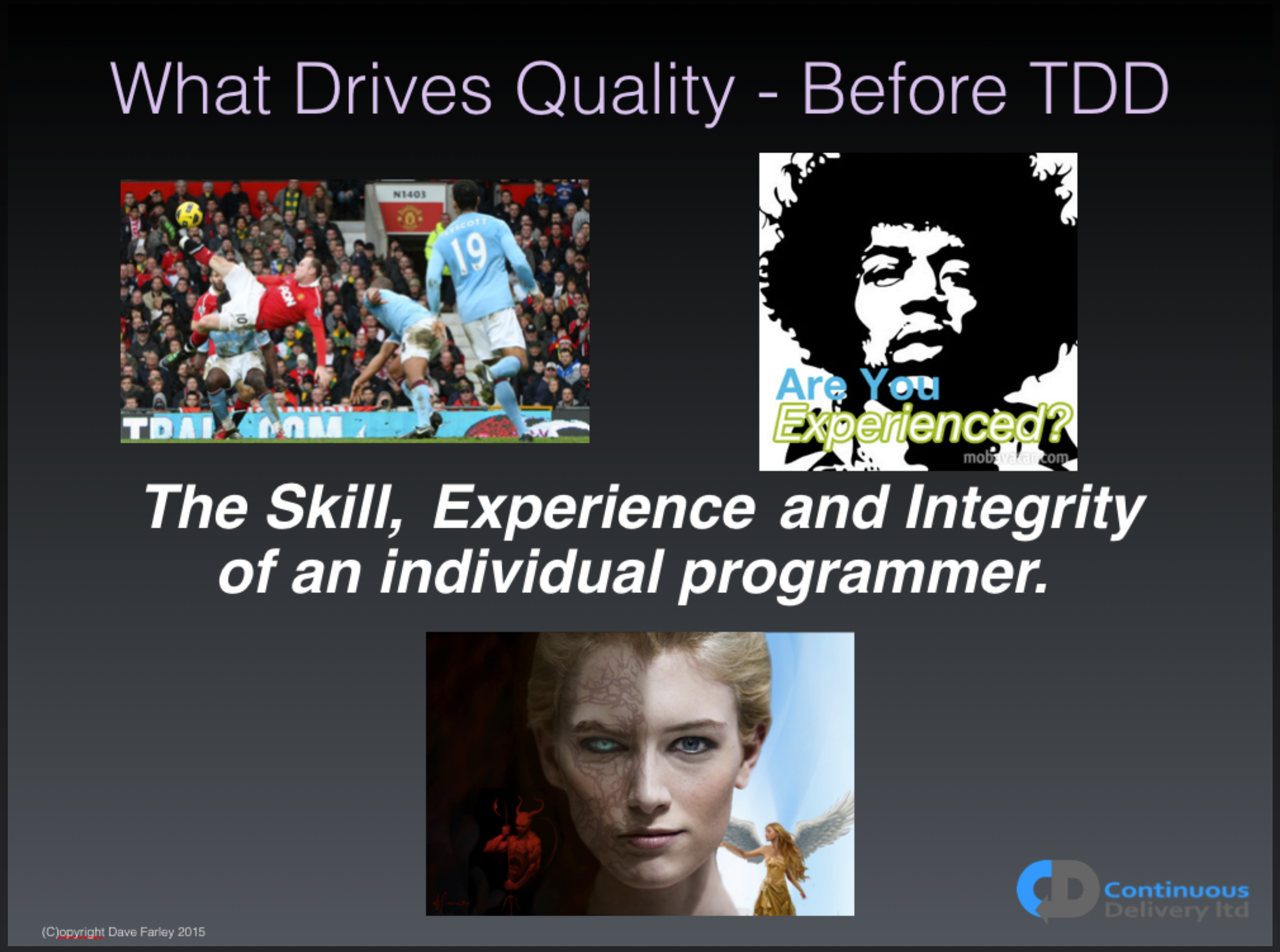
+



TDD is…

Repeat!
Red
Green
Refactor

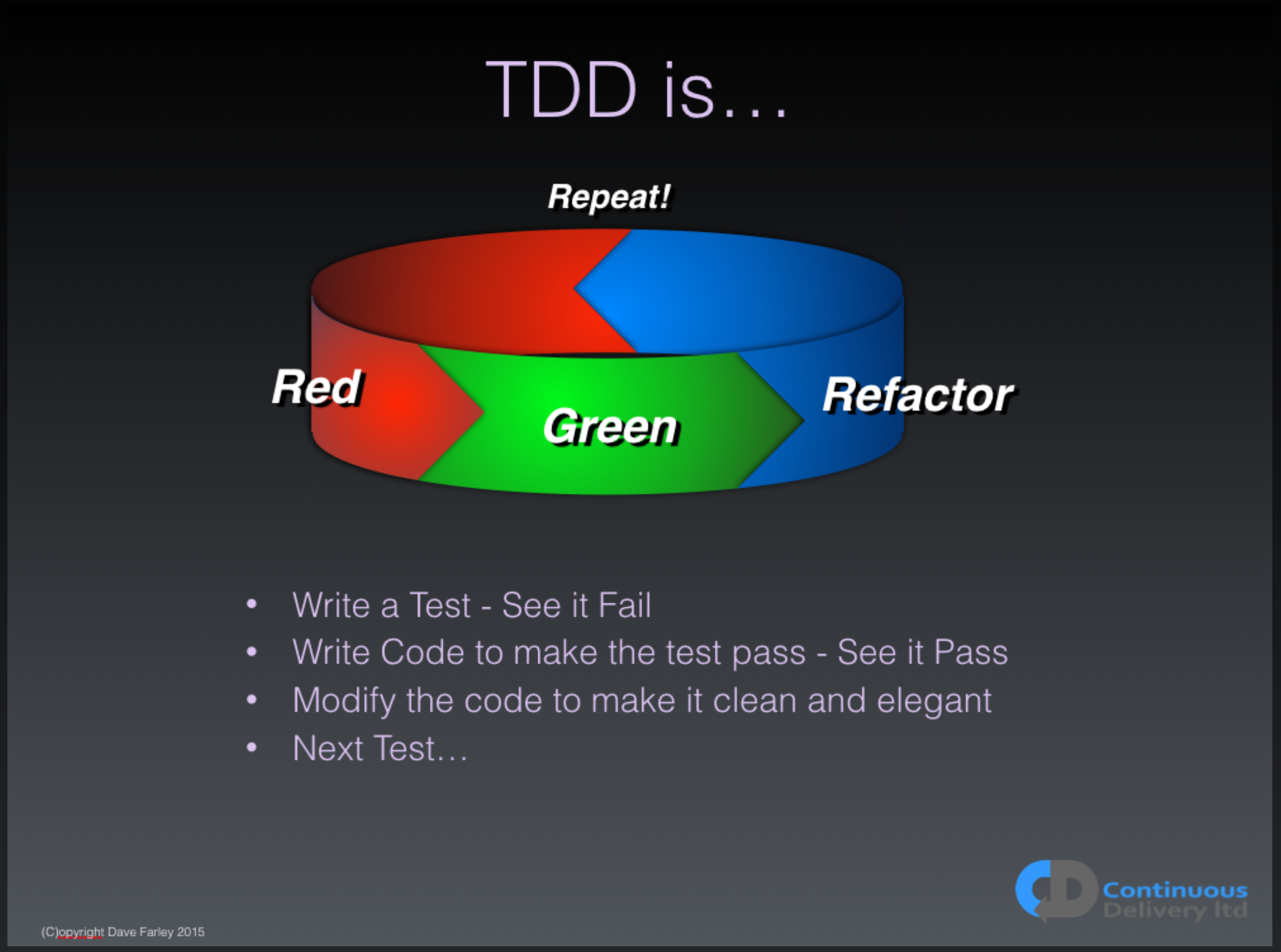
- Write a Test - See it Fail
- Write Code to make the test pass - See it Pass
- Modify the code to make it clean and elegant
- Next Test…



What Drives Quality - Before TDD

The Skill, Experience and Integrity of an individual programmer.

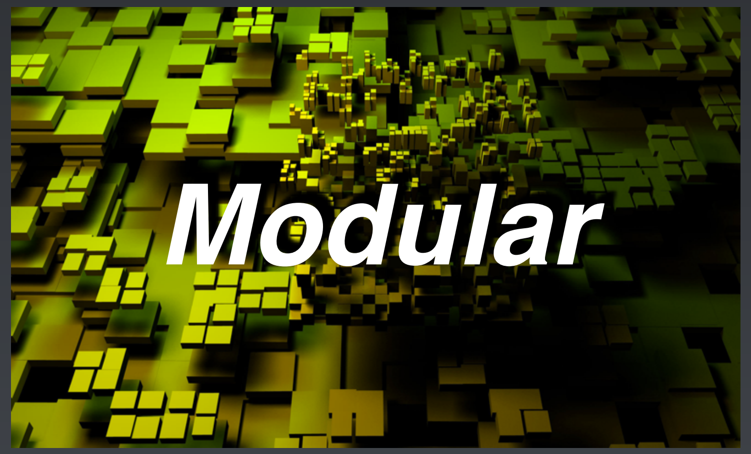
+

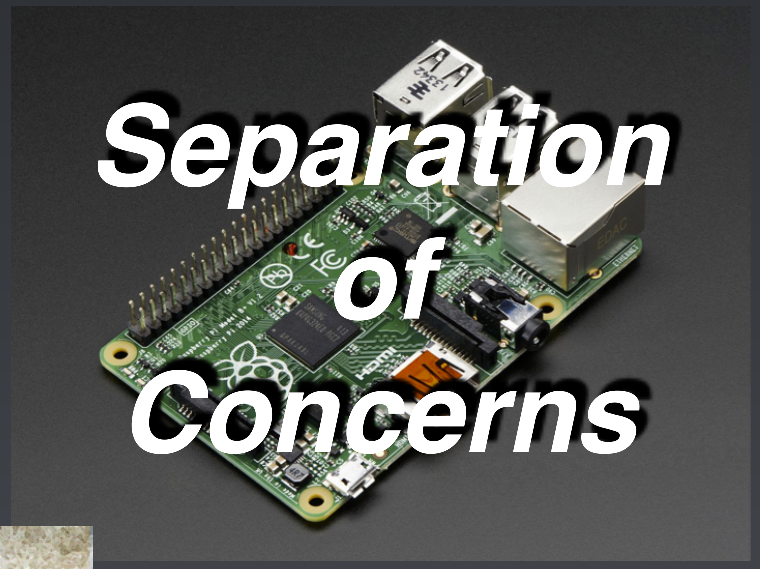


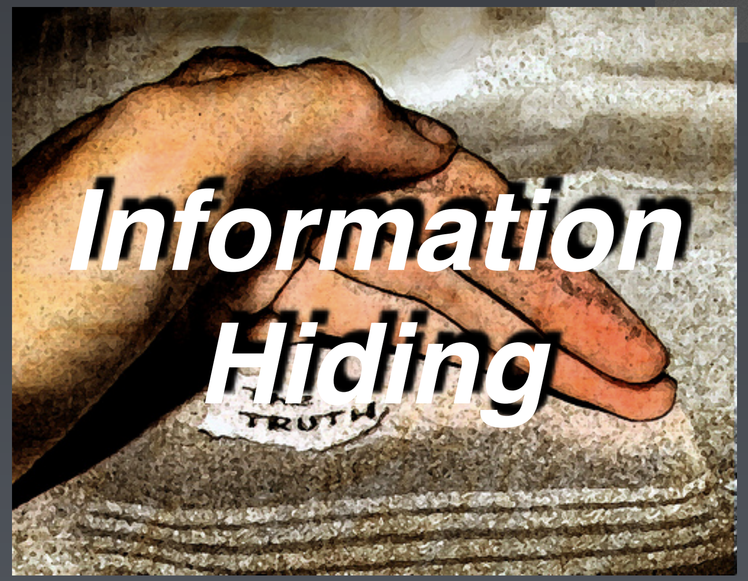
TDD is…

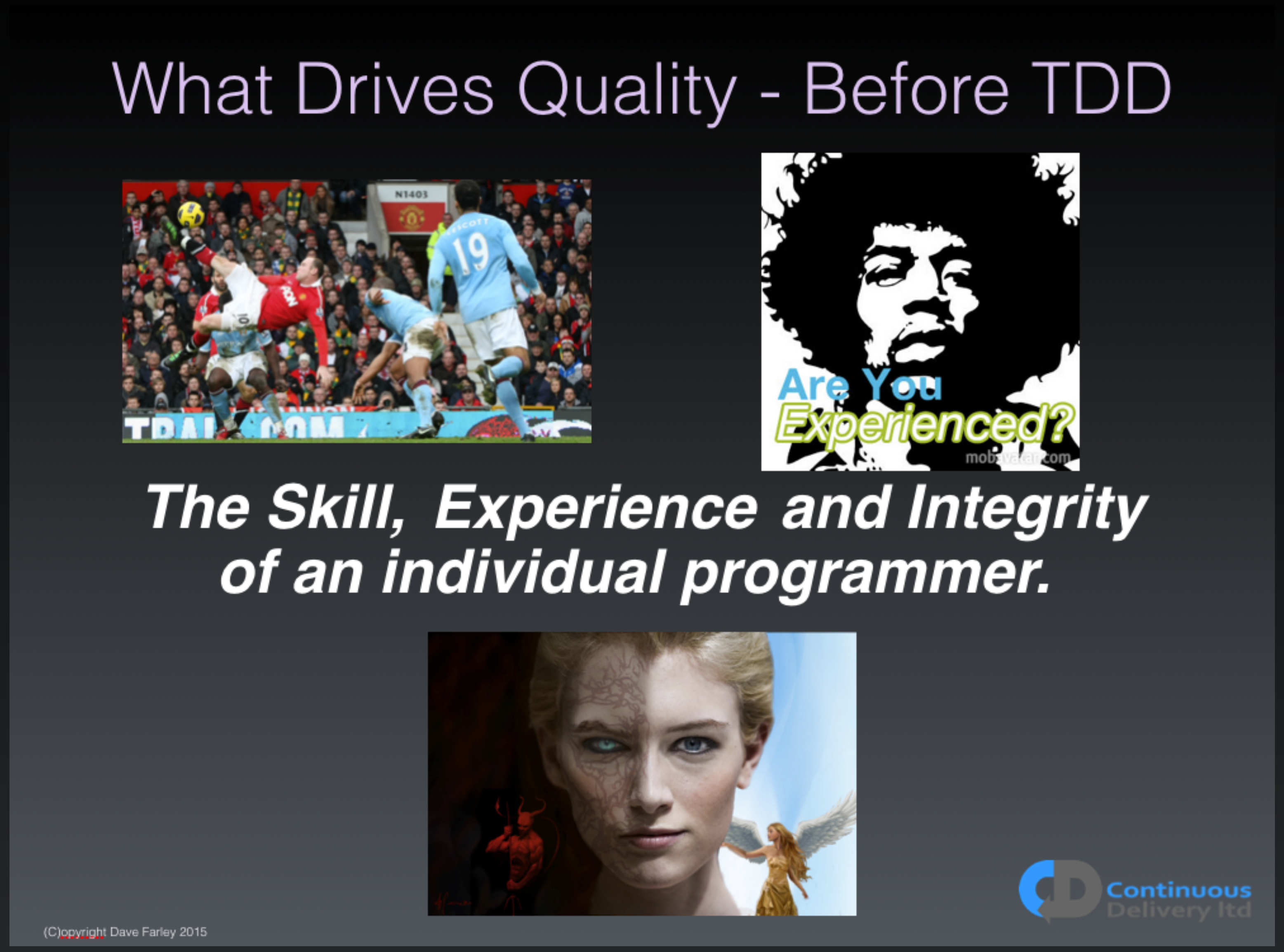
- Write a Test - See it Fail
- Write Code to make the test pass - See it Pass
- Modify the code to make it clean and elegant
- Next Test…

=





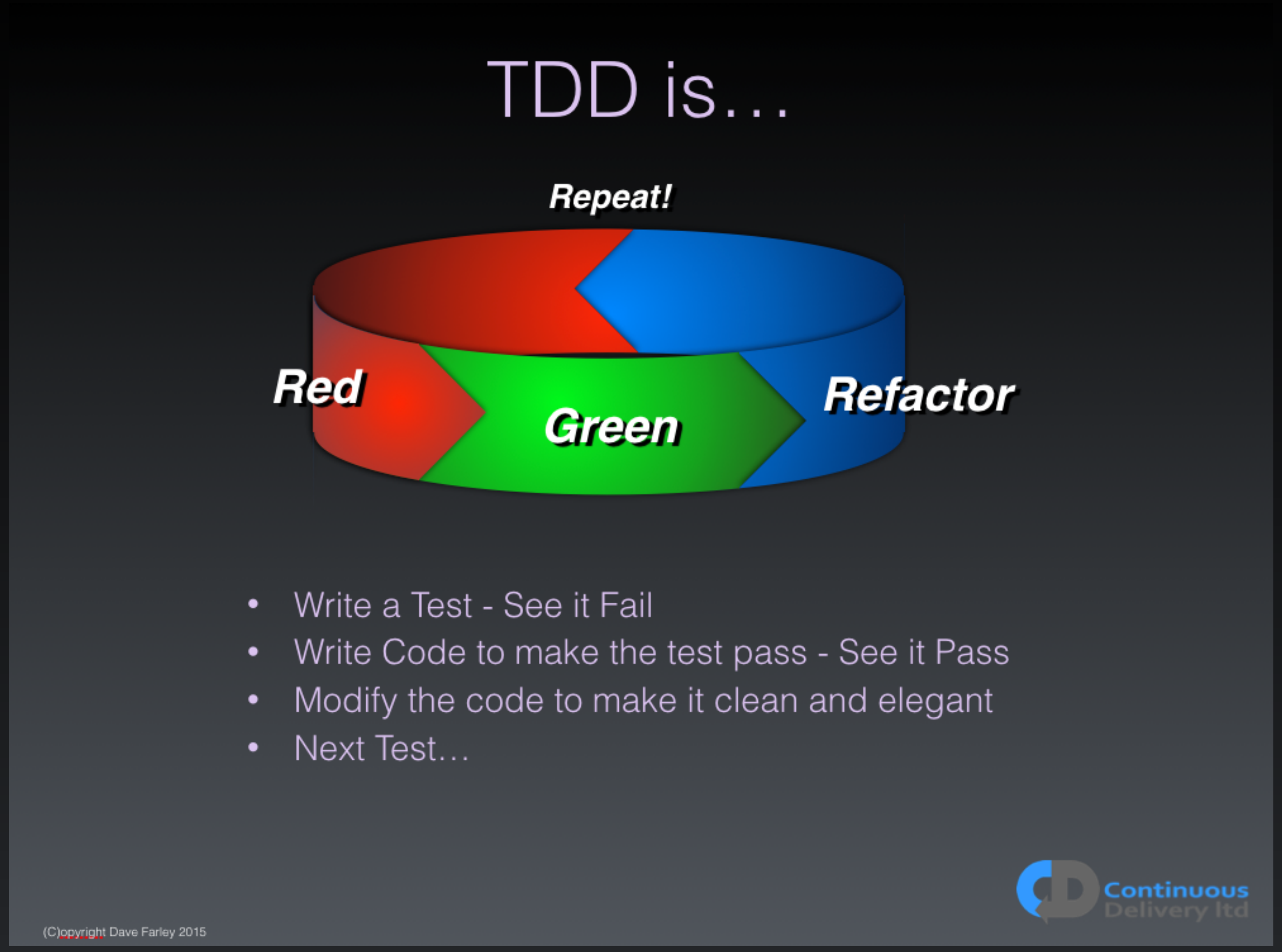




What Drives Quality - Before TDD

The Skill, Experience and Integrity of an individual programmer.

Are You Experienced?



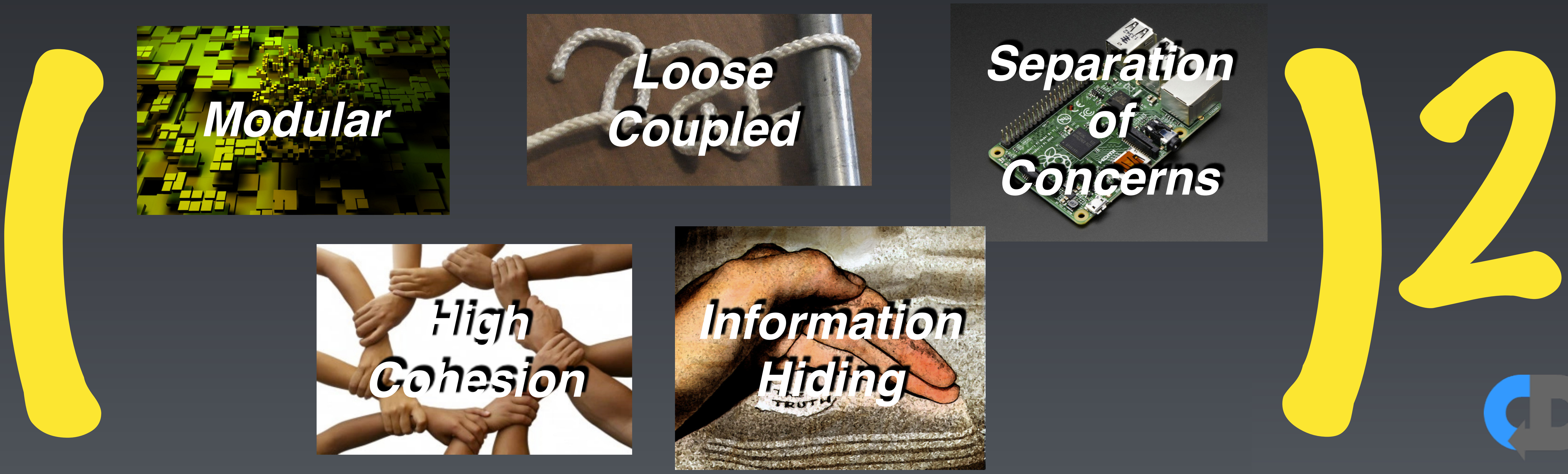
TDD is…

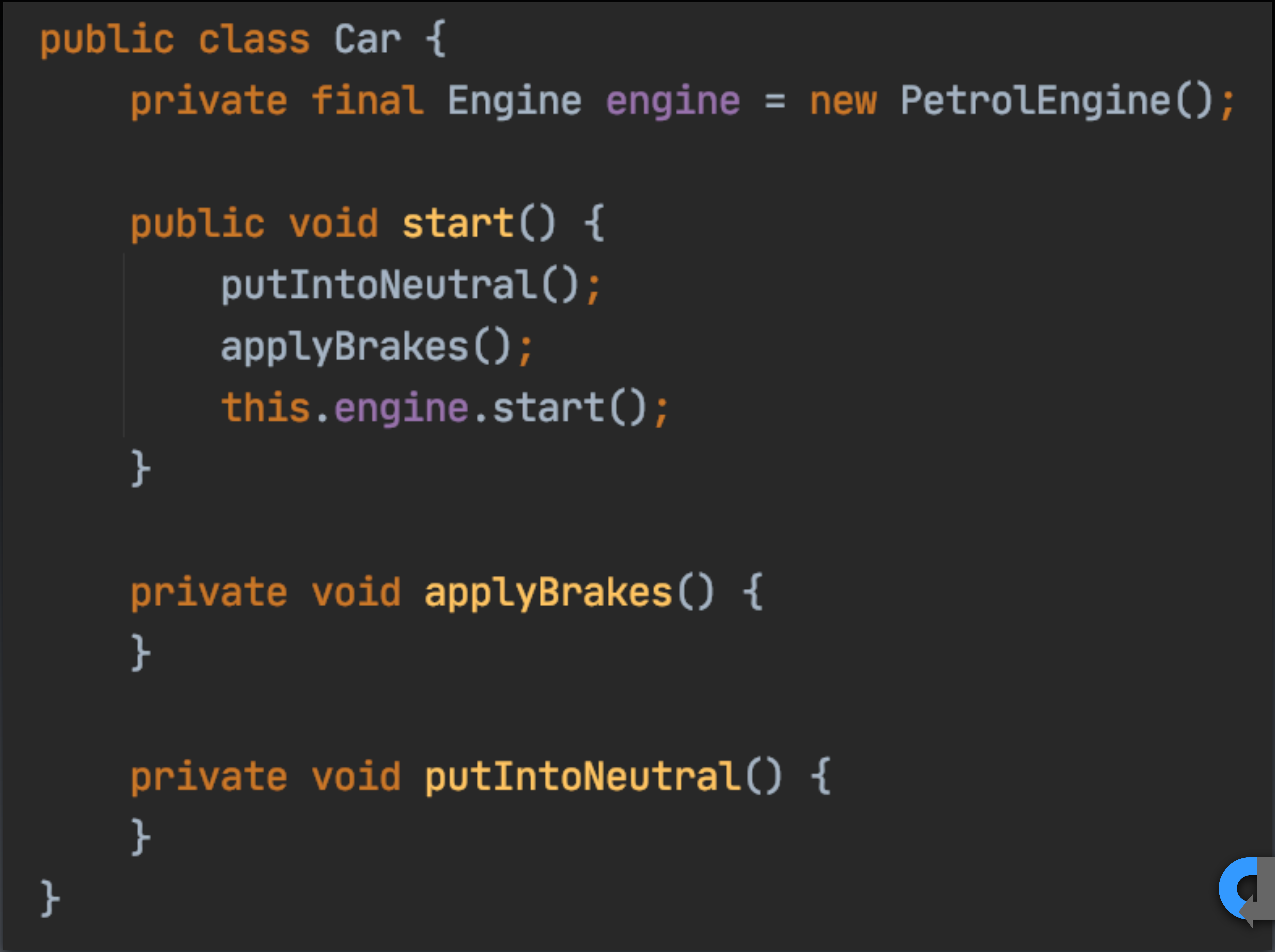
Repeat!
Red
Green
Refactor

- Write a Test - See it Fail
- Write Code to make the test pass - See it Pass
- Modify the code to make it clean and elegant
- Next Test…

+

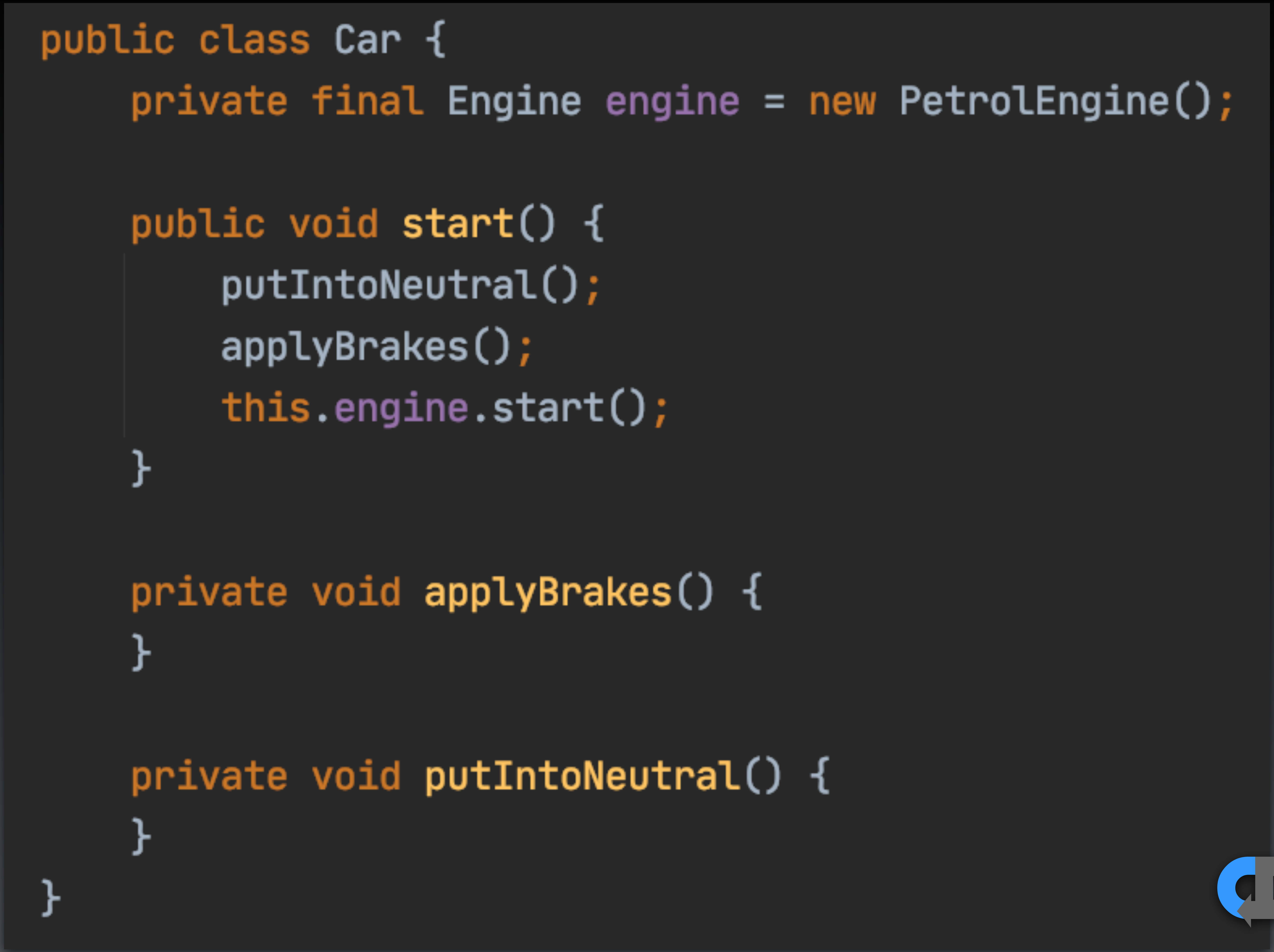
=



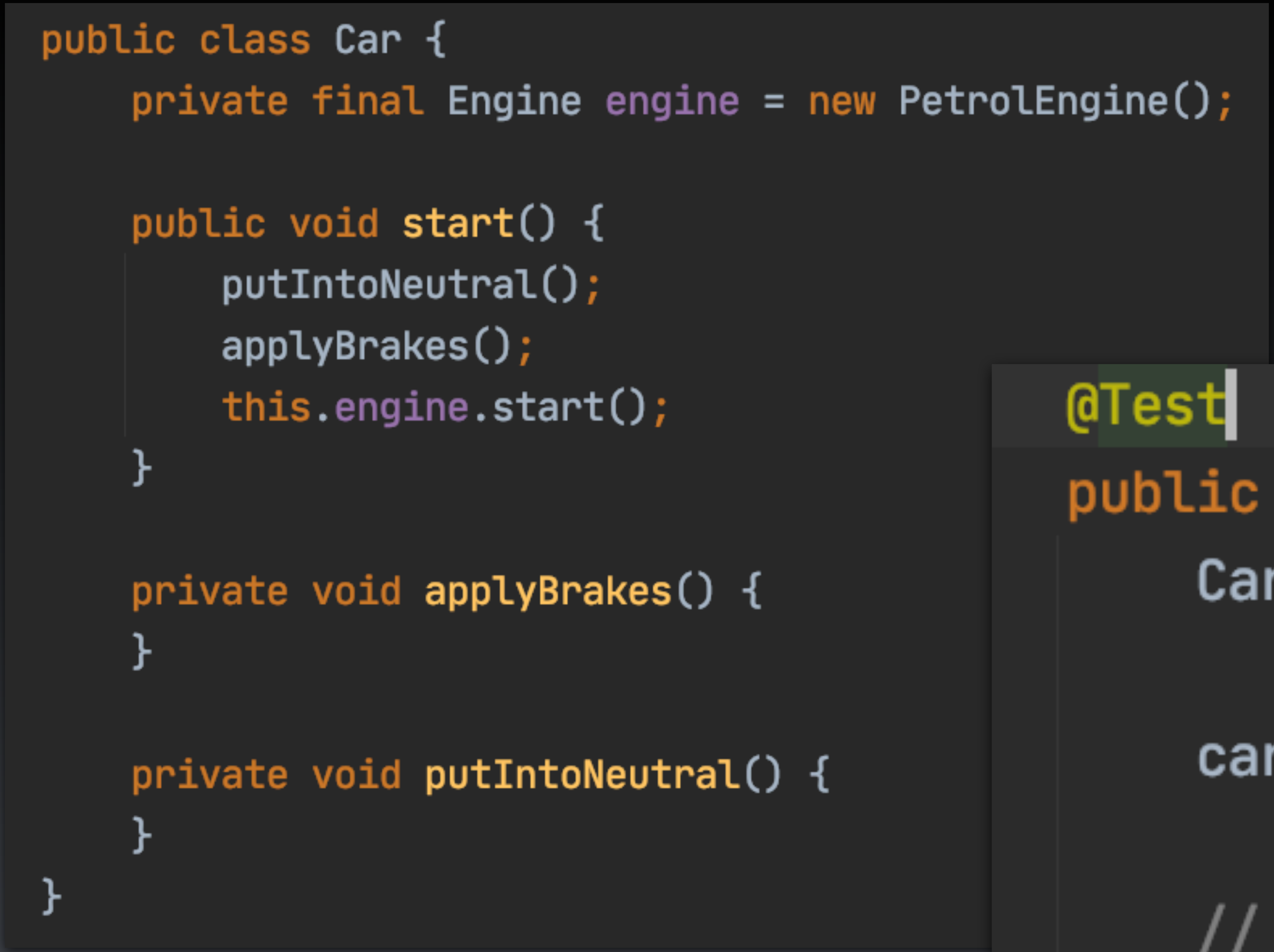


```java
public class Car {
    private final Engine engine = new PetrolEngine();

    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```

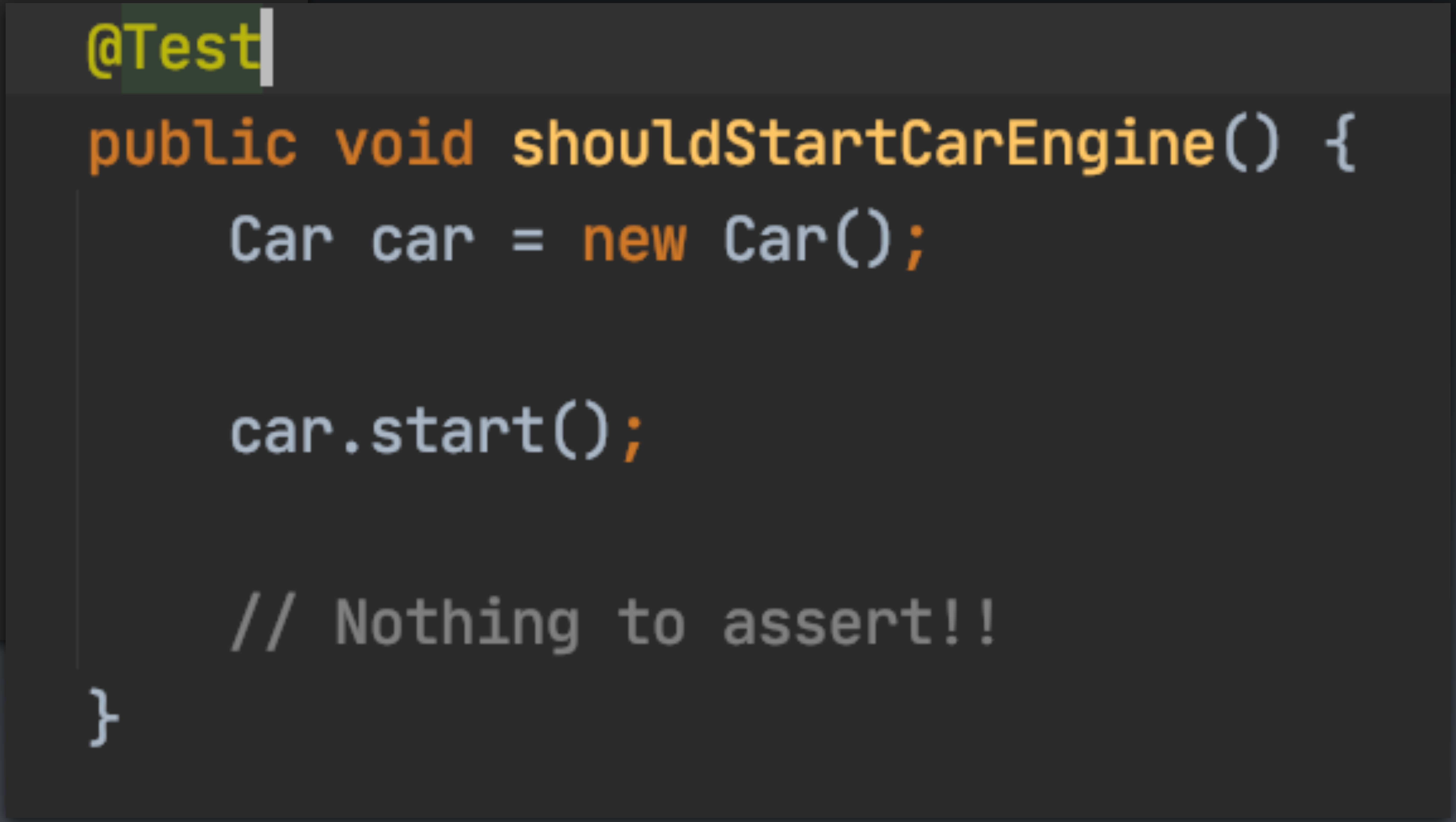
Continuous
Delivery ltd



```java
public class Car {
    private final Engine engine = new PetrolEngine();

    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```



```java
public class Car {
    private final Engine engine = new PetrolEngine();

    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```



```java
@Test
public void shouldStartCarEngine() {
    Car car = new Car();

    car.start();

    // Nothing to assert!!
}
```

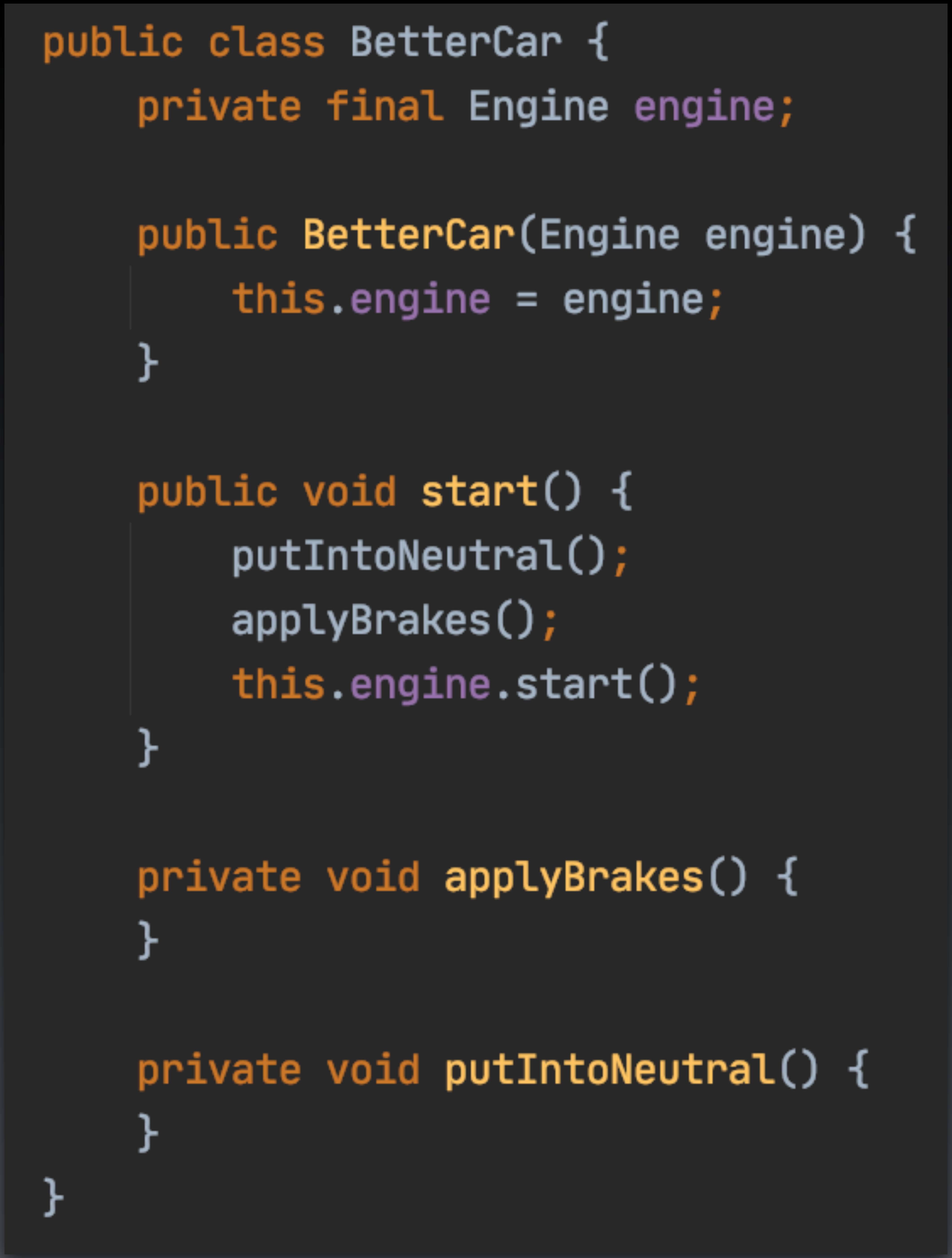
Continuous Delivery ltd

```java
public class Car {
    private final Engine engine = new PetrolEngine();

    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```

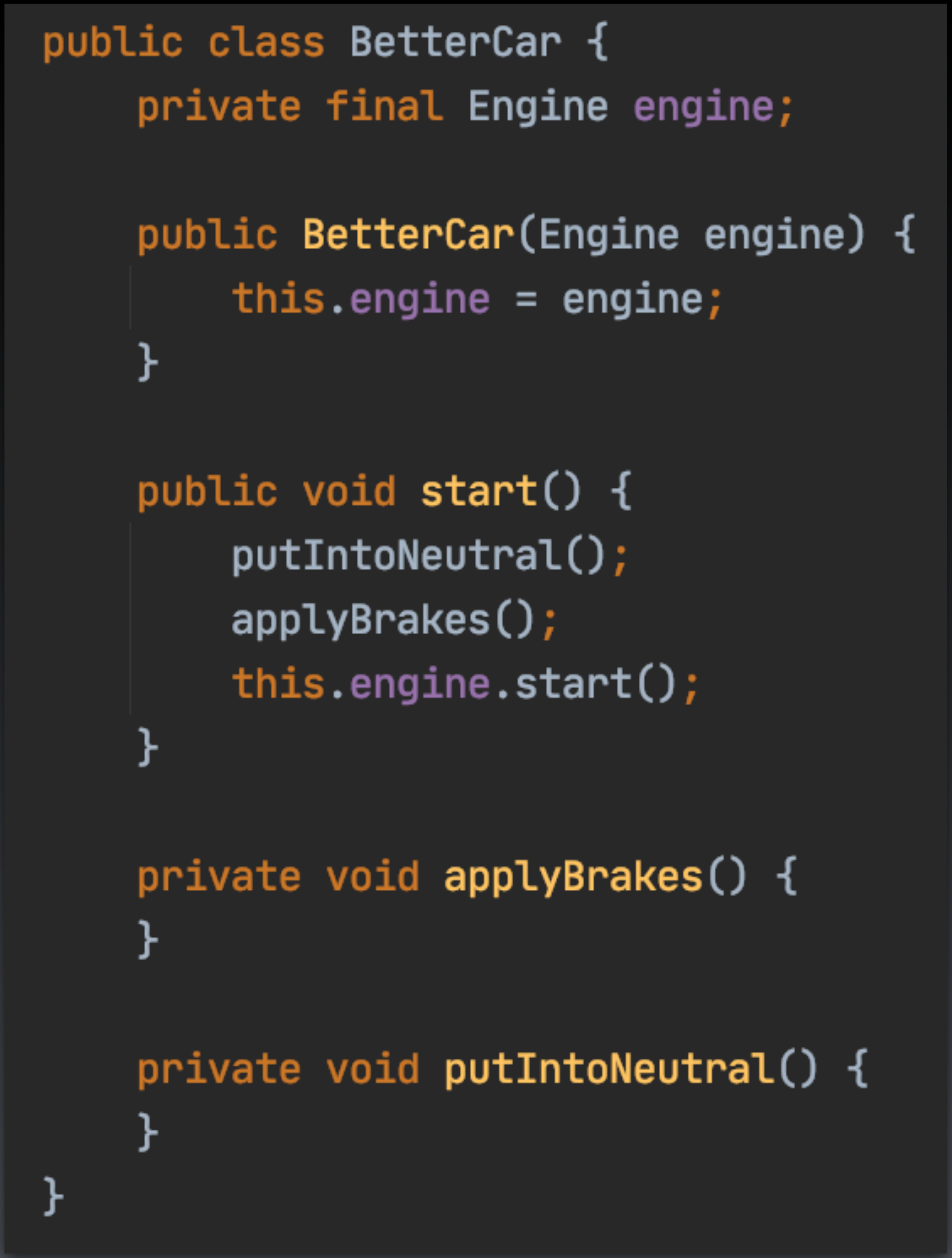
```java
@Test
public void shouldStartCarEngine() {
    Car car = new Car();

    car.start();

    // Nothing to assert!!
}
```

Continuous Delivery ltd

Continuous Delivery ltd

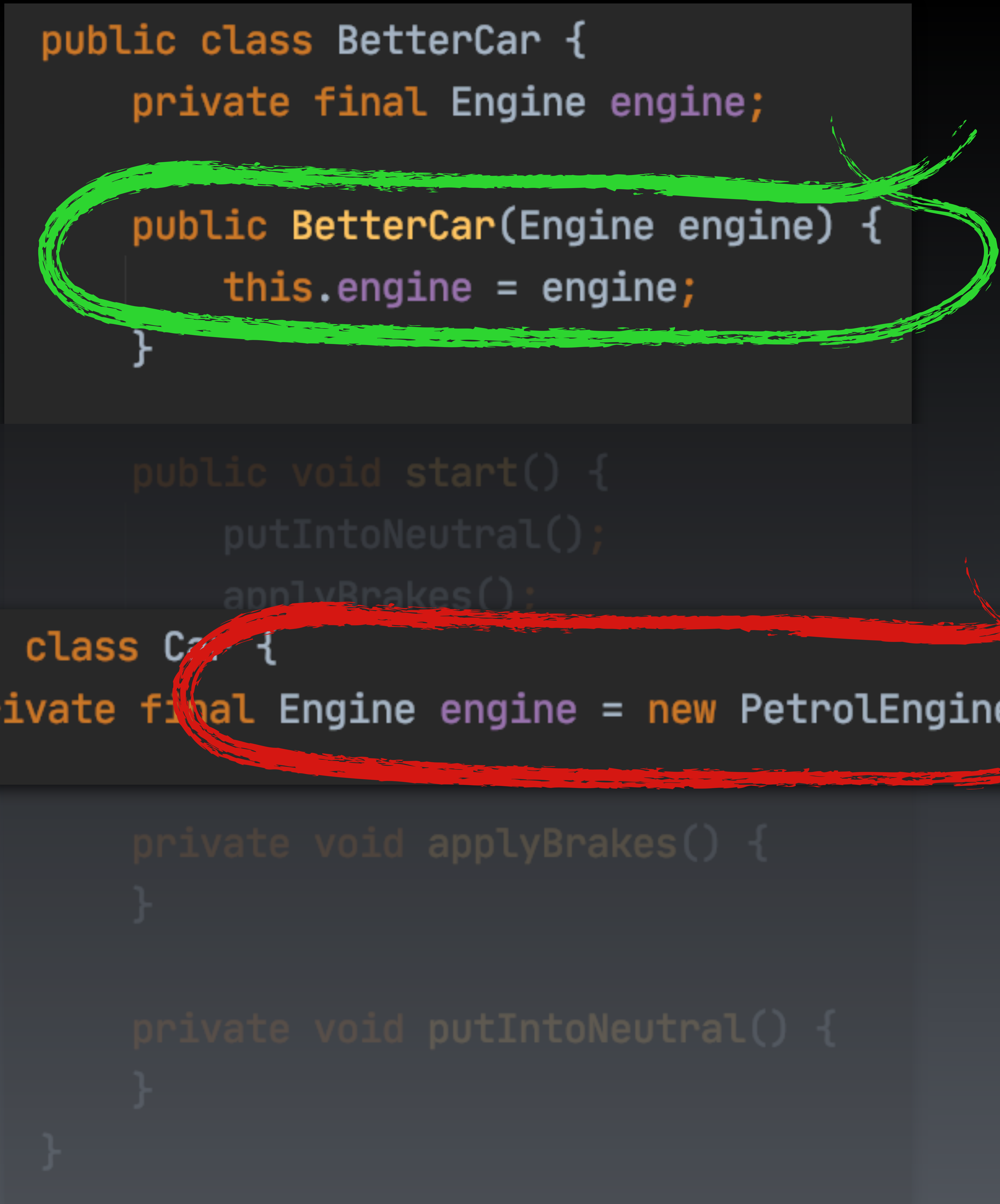


```java
public class BetterCar {
    private final Engine engine;

    public BetterCar(Engine engine) {
        this.engine = engine;
    }

    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```

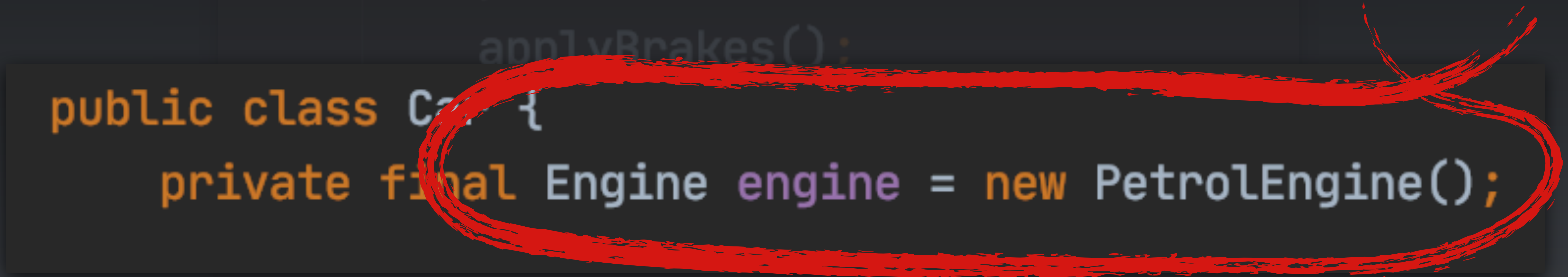


```java
public class BetterCar {
    private final Engine engine;

    public BetterCar(Engine engine) {
        this.engine = engine;
    }

    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```

Continuous
Delivery ltd

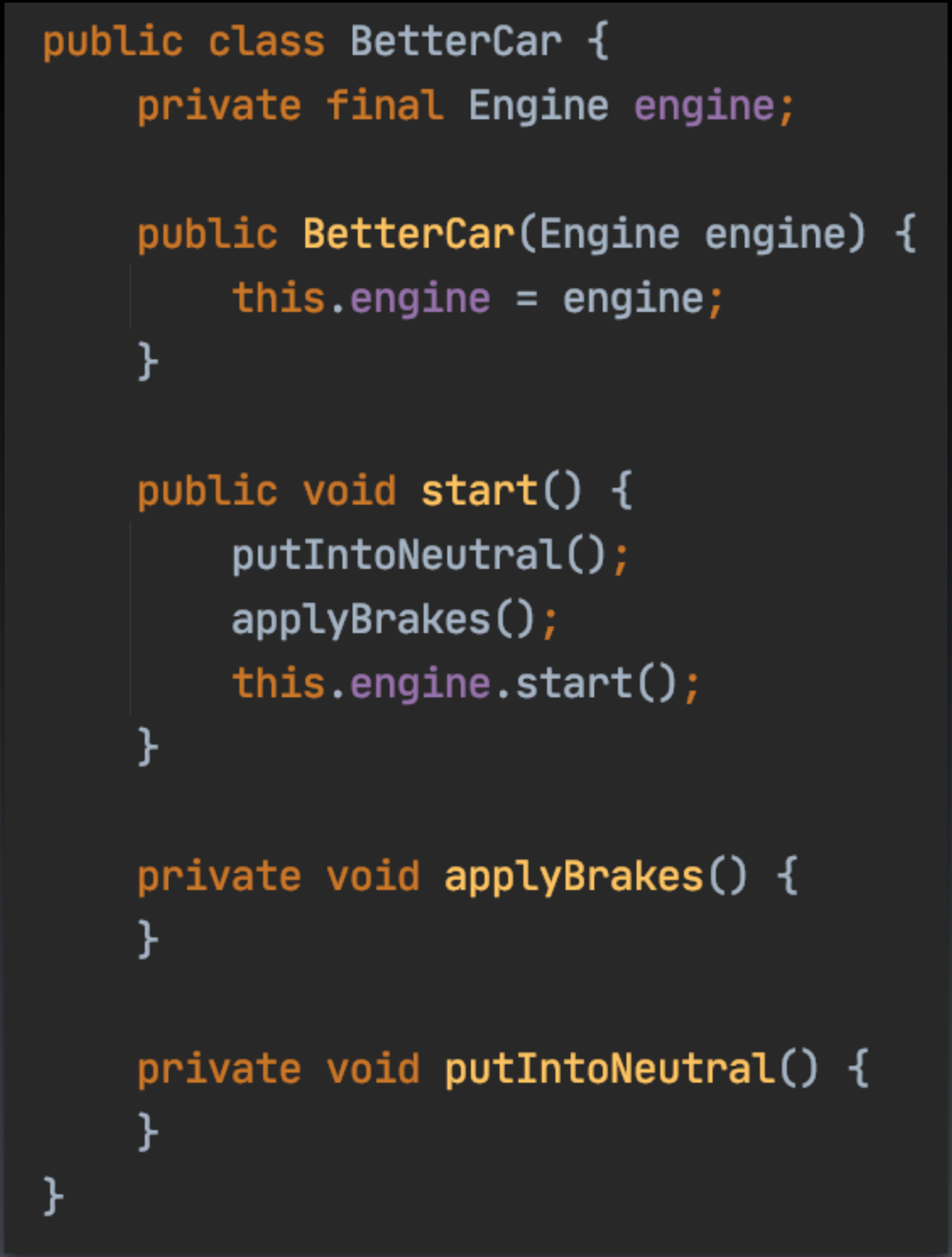


```java
public class BetterCar {
    private final Engine engine;

    public BetterCar(Engine engine) {
        this.engine = engine;
    }

    public void start() {
        putIntoNeutral();
        applyBrakes();

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```



```java
public class Car {
    private final Engine engine = new PetrolEngine();
```

Continuous
Delivery ltd

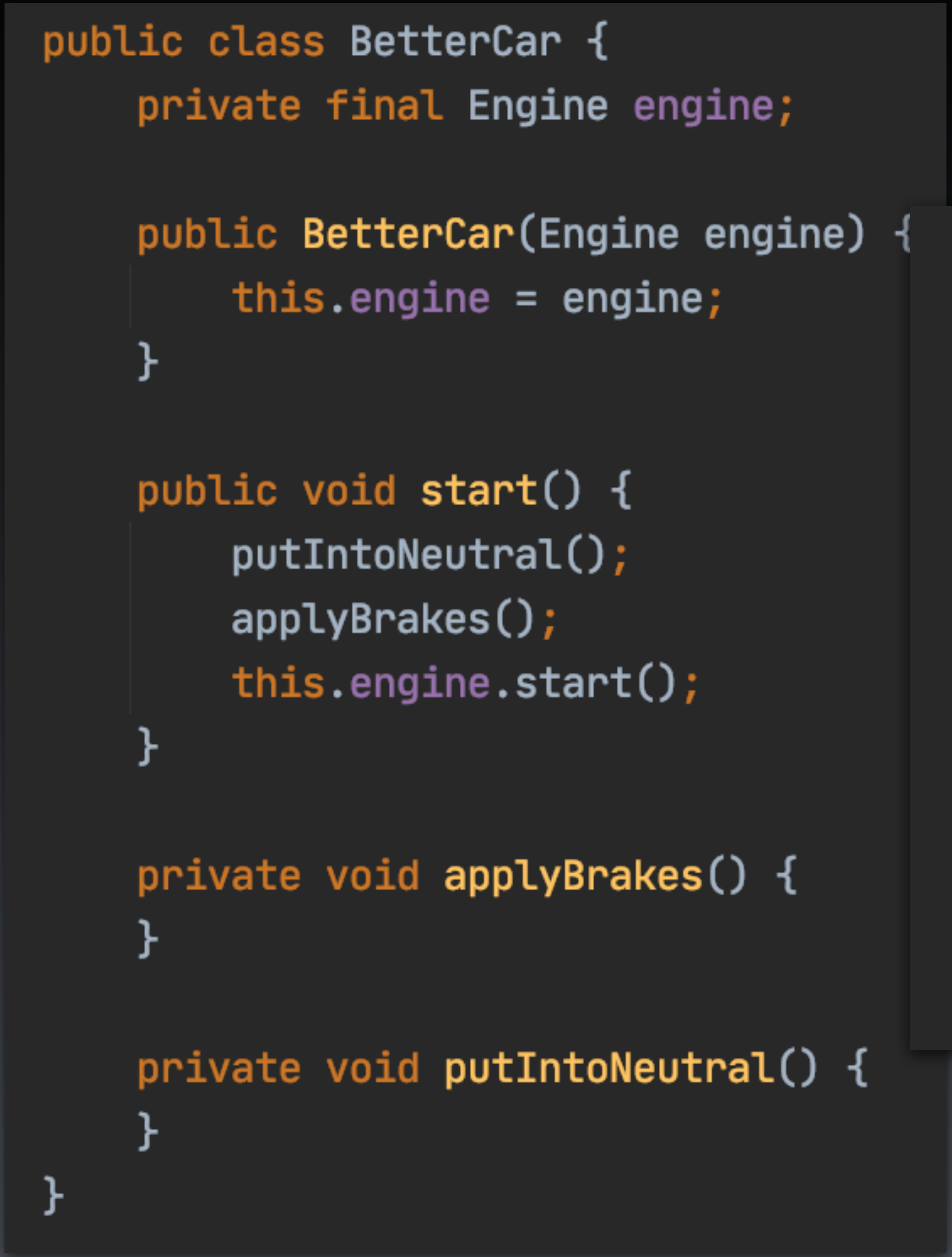


```java
public class BetterCar {
    private final Engine engine;

    public BetterCar(Engine engine) {
        this.engine = engine;
    }

    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```

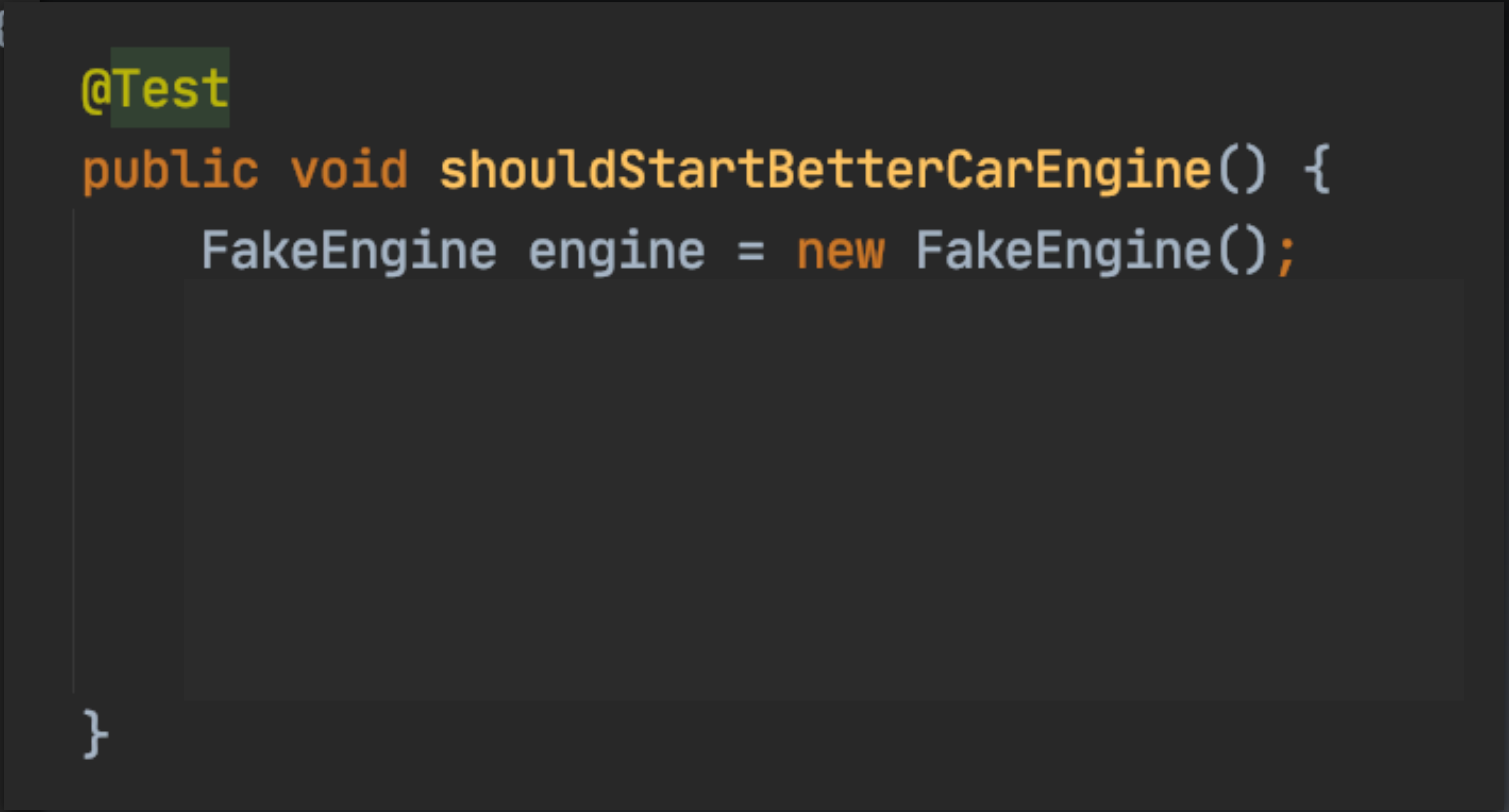
```java
public class BetterCar {
    private final Engine engine;

    public BetterCar(Engine engine) {
        this.engine = engine;
    }

    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```

```java
public class BetterCar {
    private final Engine engine;

    public BetterCar(Engine engine) {
        this.engine = engine;
    }

    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```

```java
@Test
public void shouldStartBetterCarEngine() {

}
```

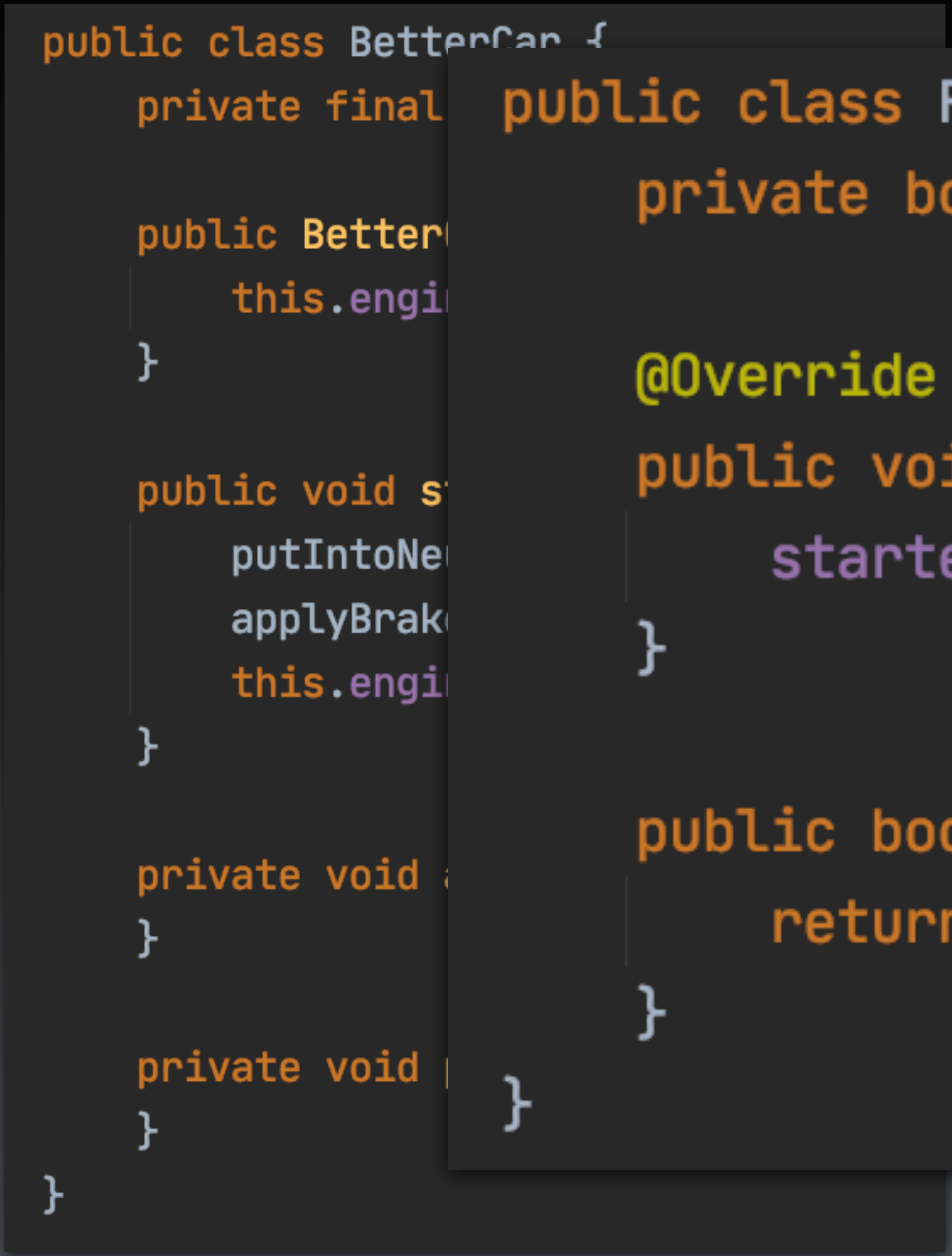


```java
public class BetterCar {
    private final Engine engine;

    public BetterCar(Engine engine) {
        this.engine = engine;
    }

    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```

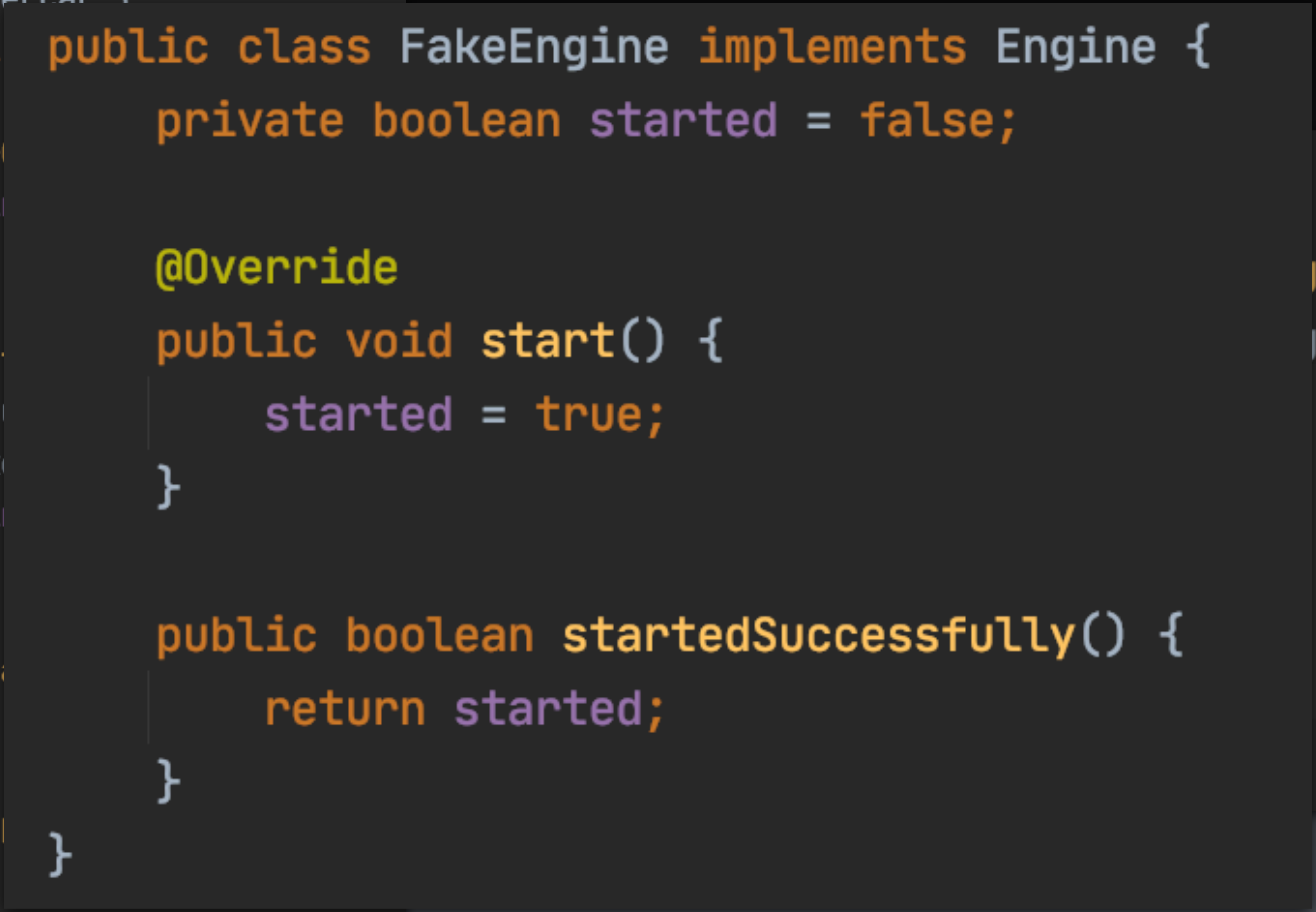


```java
@Test
public void shouldStartBetterCarEngine() {
    FakeEngine engine = new FakeEngine();



}
```

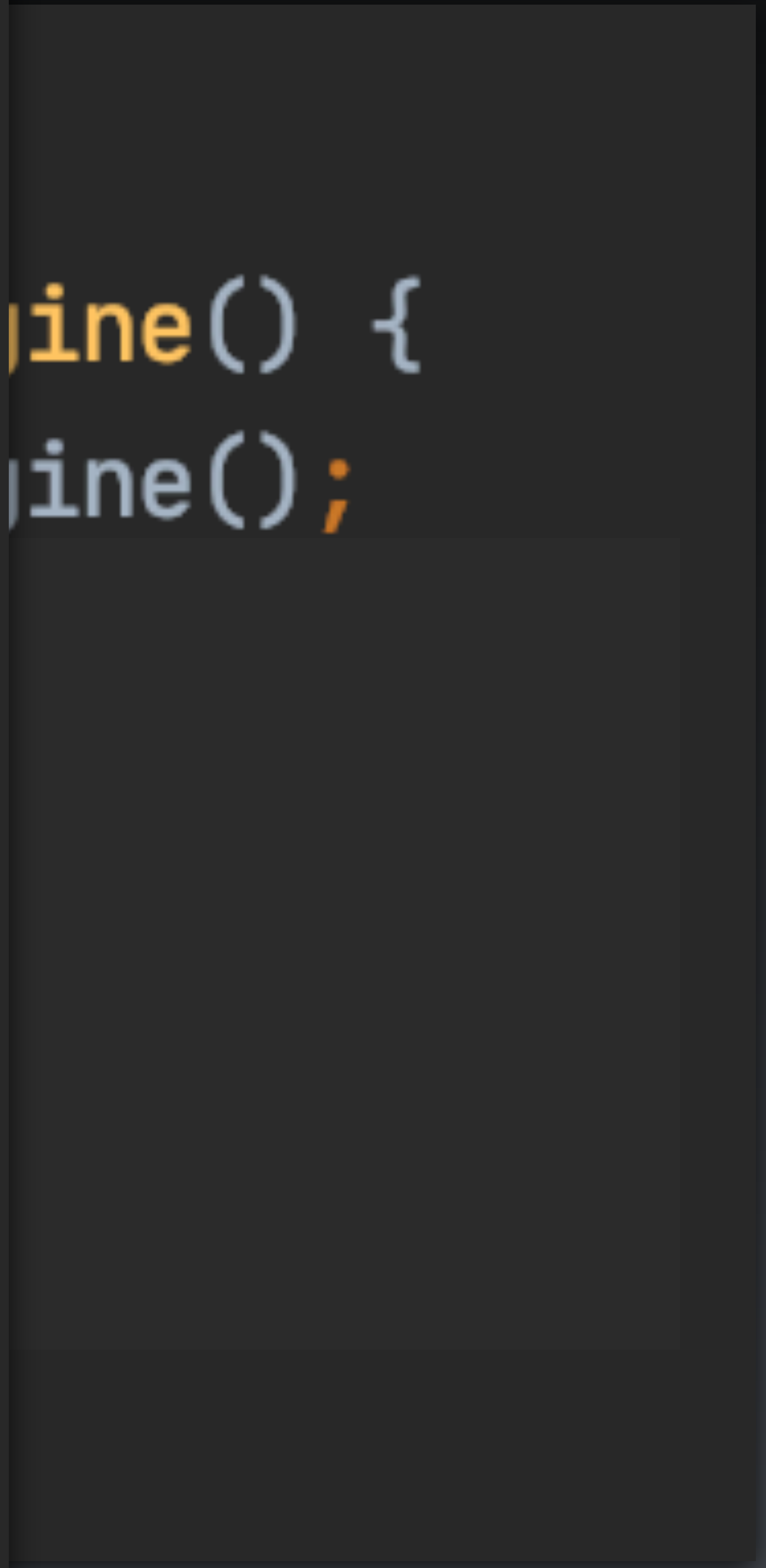
Continuous
Delivery ltd



```
public class BetterCar {
    private final

    public Better
        this.engi
    }

    public void s
        putIntoNe
        applyBrak
        this.engi
    }

    private void
    }

    private void
    }
}
```

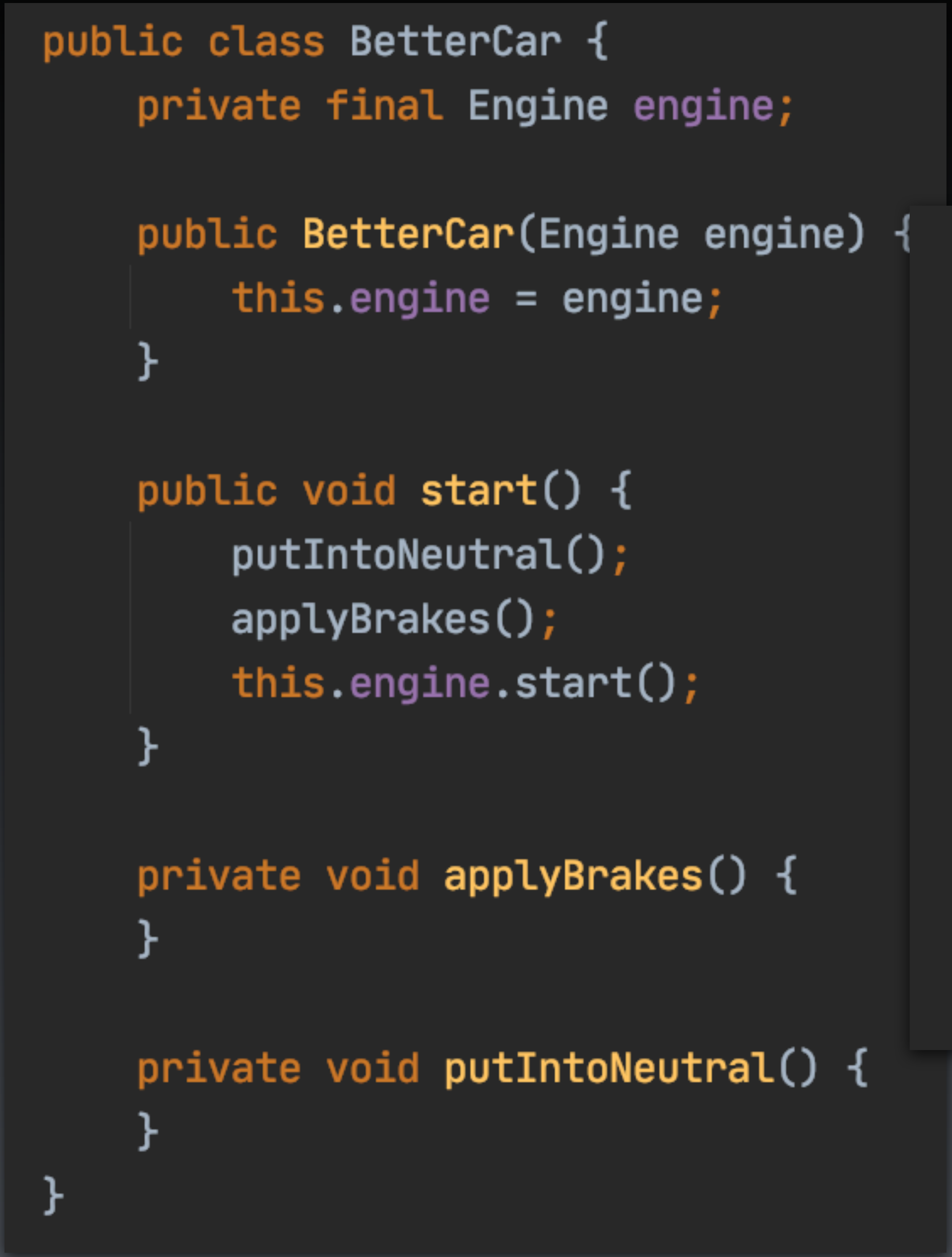


```
public class FakeEngine implements Engine {
    private boolean started = false;

    @Override
    public void start() {
        started = true;
    }

    public boolean startedSuccessfully() {
        return started;
    }
}
```

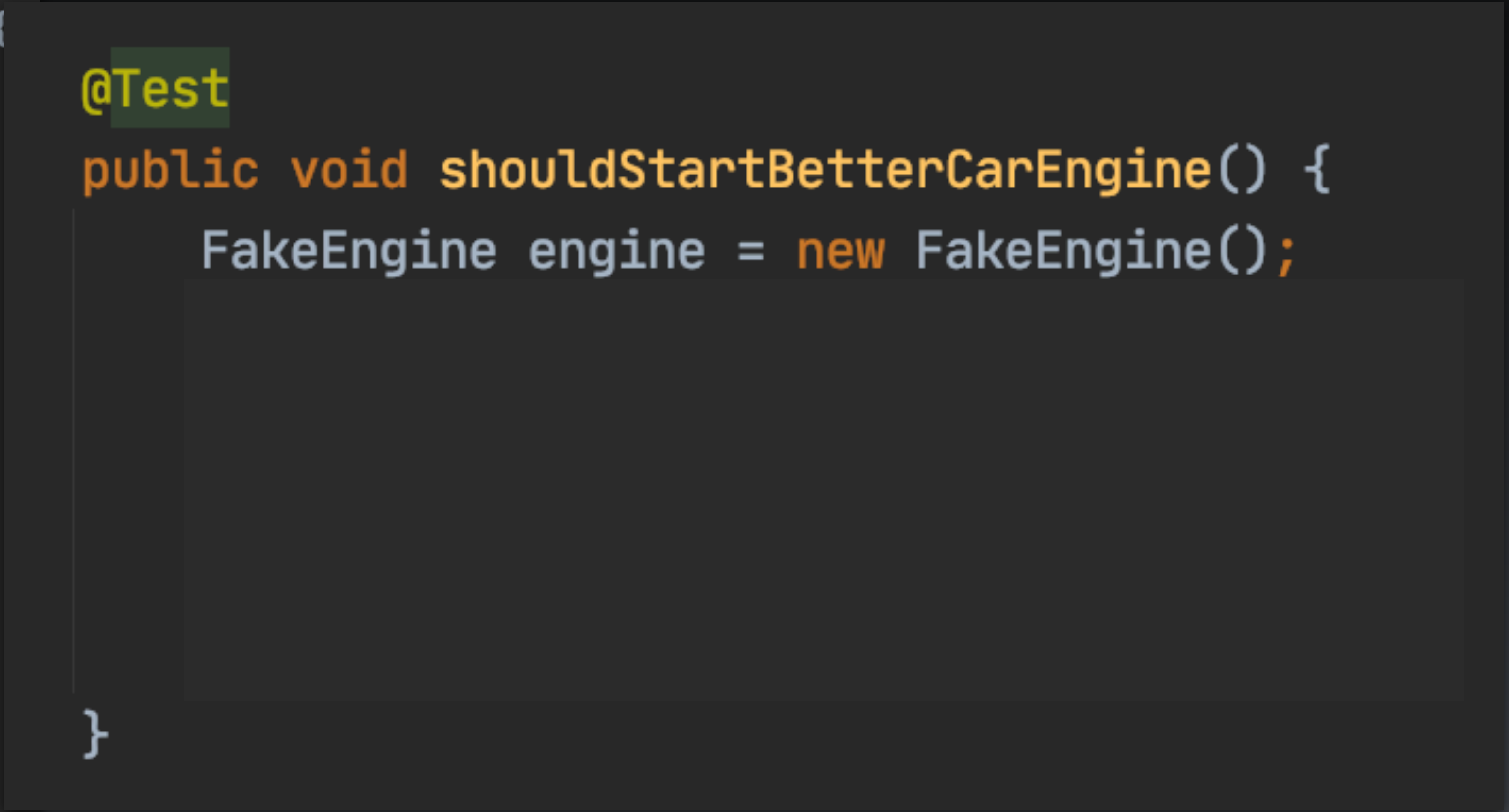


```
ine() {
ine();
```

Continuous Delivery ltd

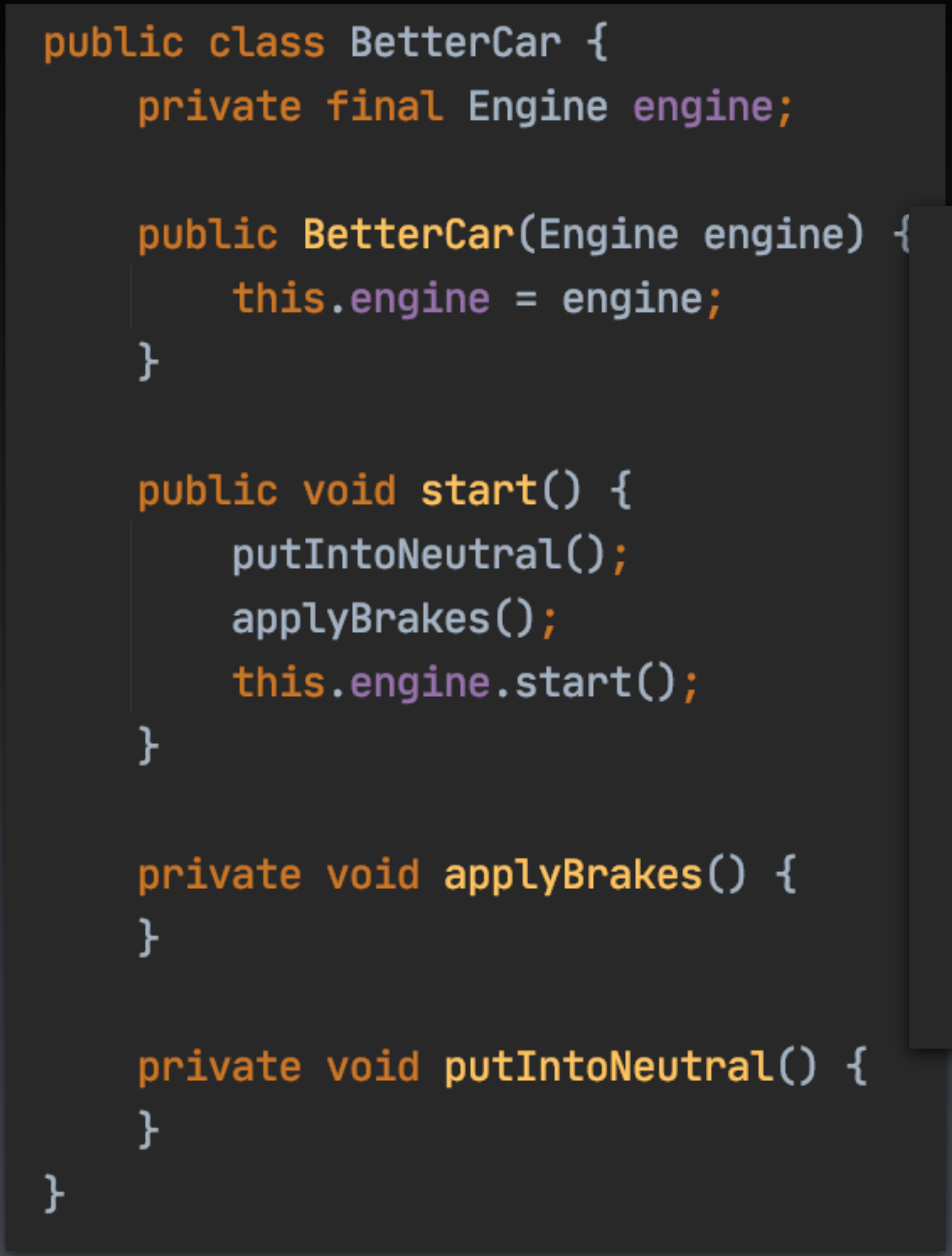


```java
public class BetterCar {
    private final Engine engine;

    public BetterCar(Engine engine) {
        this.engine = engine;
    }

    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```

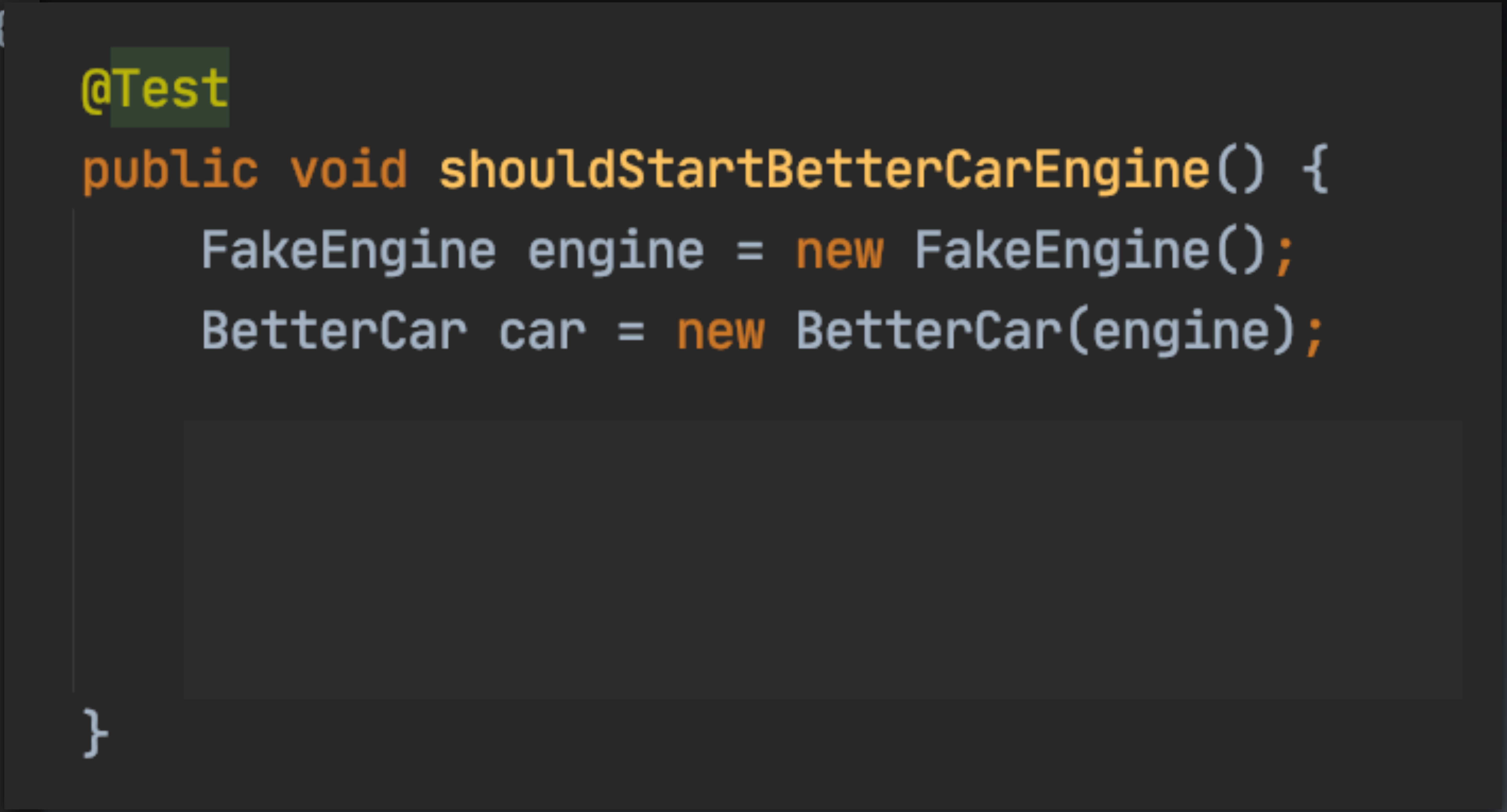


```java
@Test
public void shouldStartBetterCarEngine() {
    FakeEngine engine = new FakeEngine();


}
```

Continuous Delivery ltd

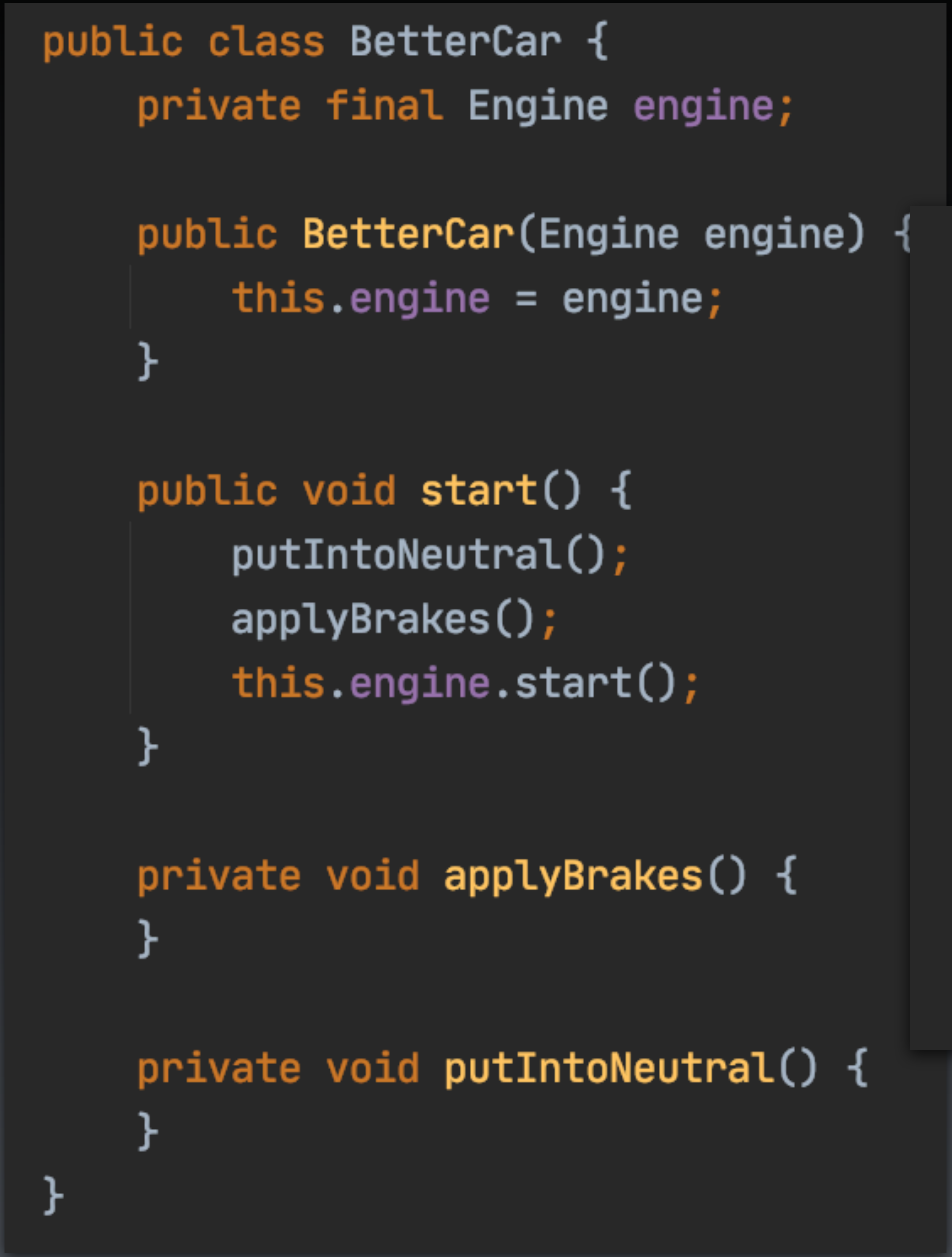


```java
public class BetterCar {
    private final Engine engine;

    public BetterCar(Engine engine) {
        this.engine = engine;
    }

    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```

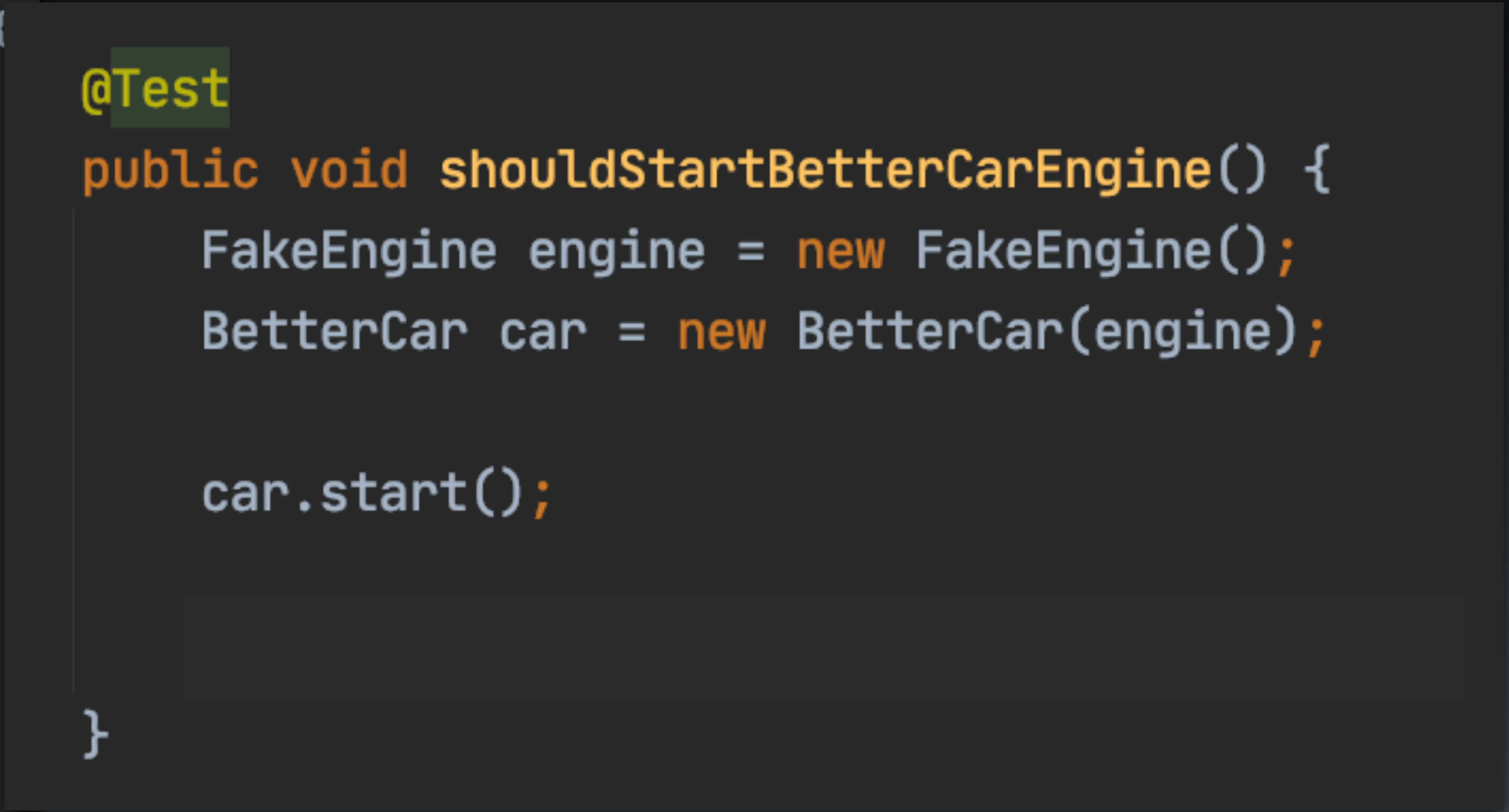


```java
@Test
public void shouldStartBetterCarEngine() {
    FakeEngine engine = new FakeEngine();
    BetterCar car = new BetterCar(engine);


}
```

Continuous
Delivery ltd

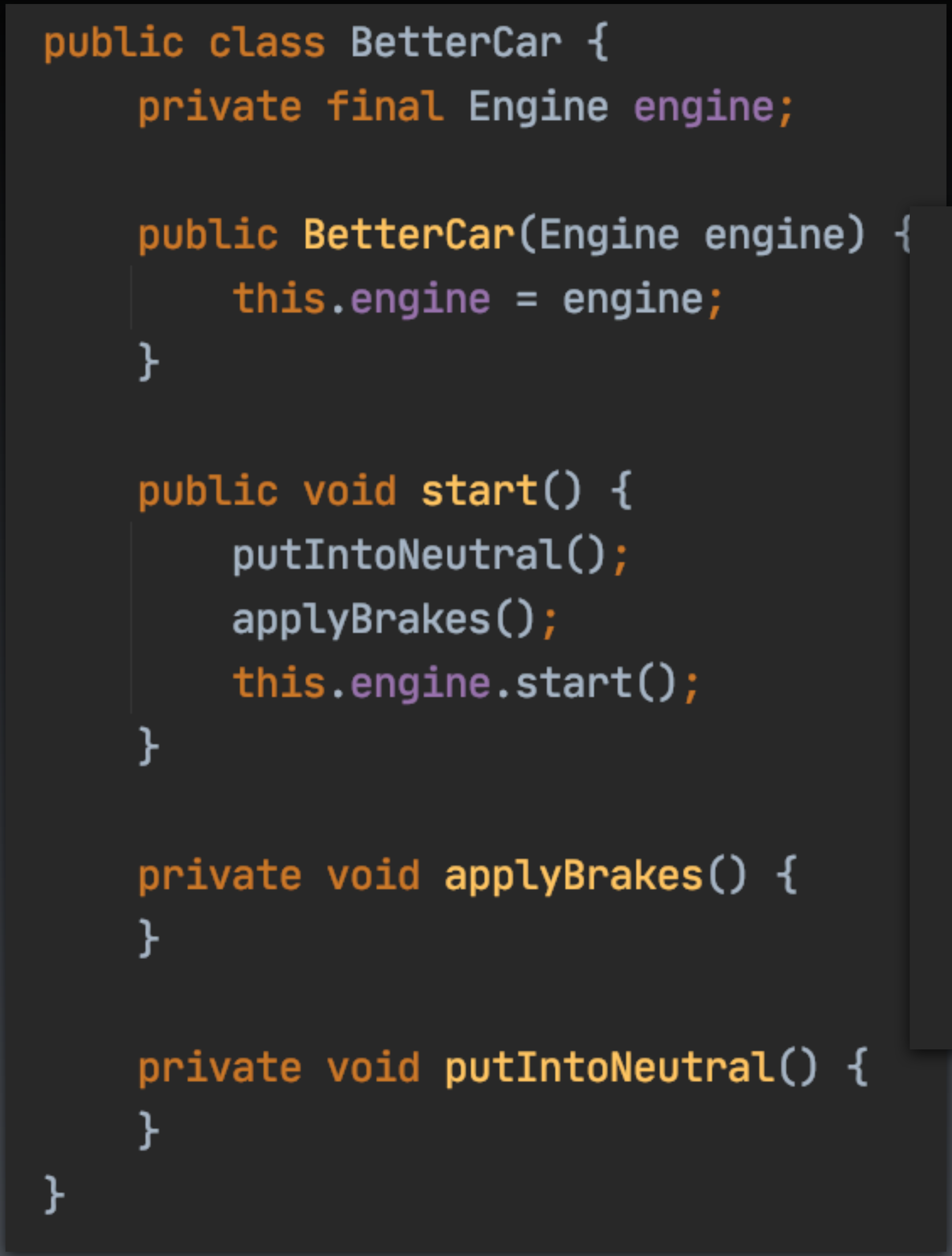


```java
public class BetterCar {
    private final Engine engine;

    public BetterCar(Engine engine) {
        this.engine = engine;
    }

    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```

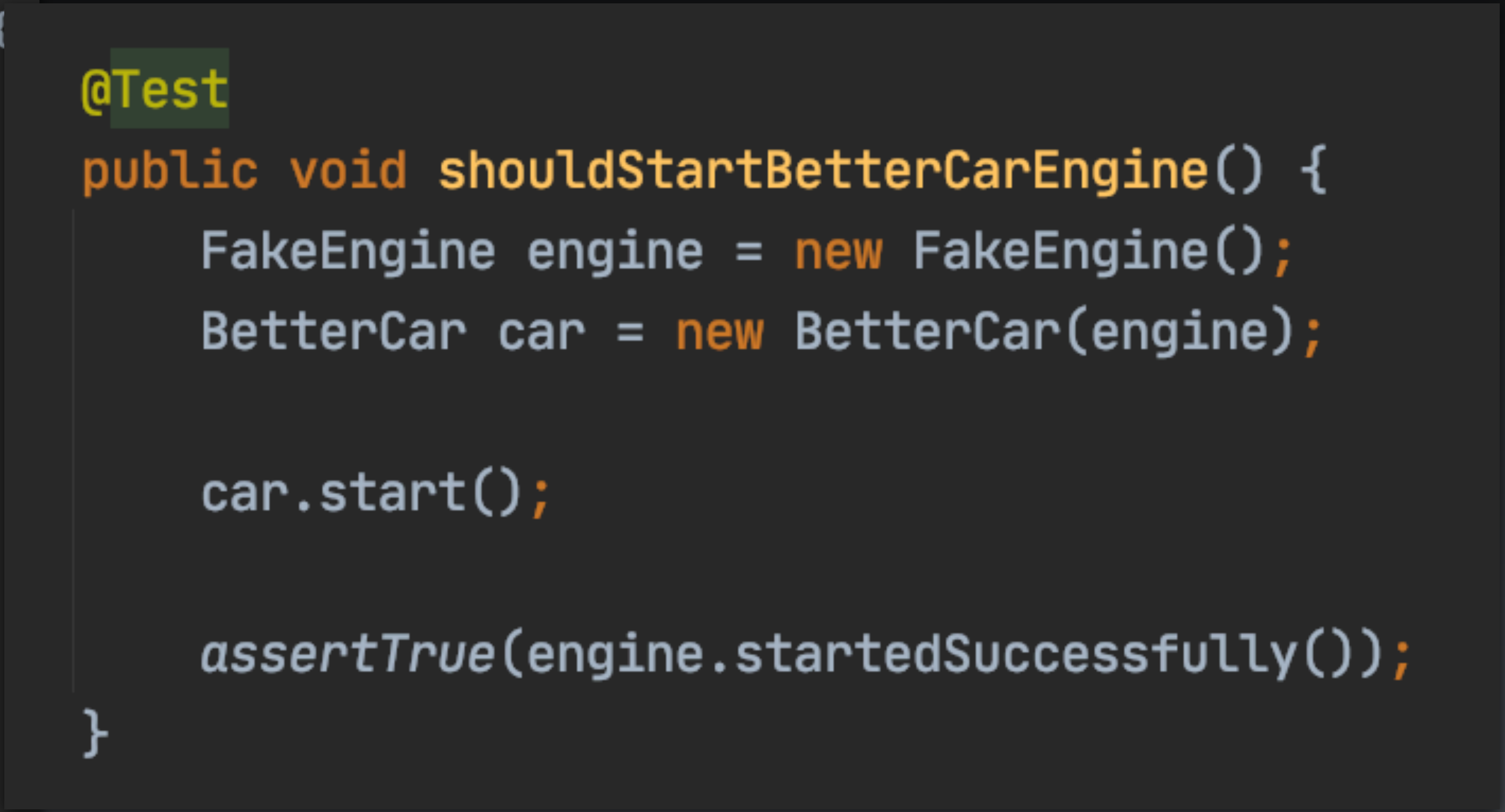


```java
@Test
public void shouldStartBetterCarEngine() {
    FakeEngine engine = new FakeEngine();
    BetterCar car = new BetterCar(engine);

    car.start();


}
```

Continuous Delivery ltd

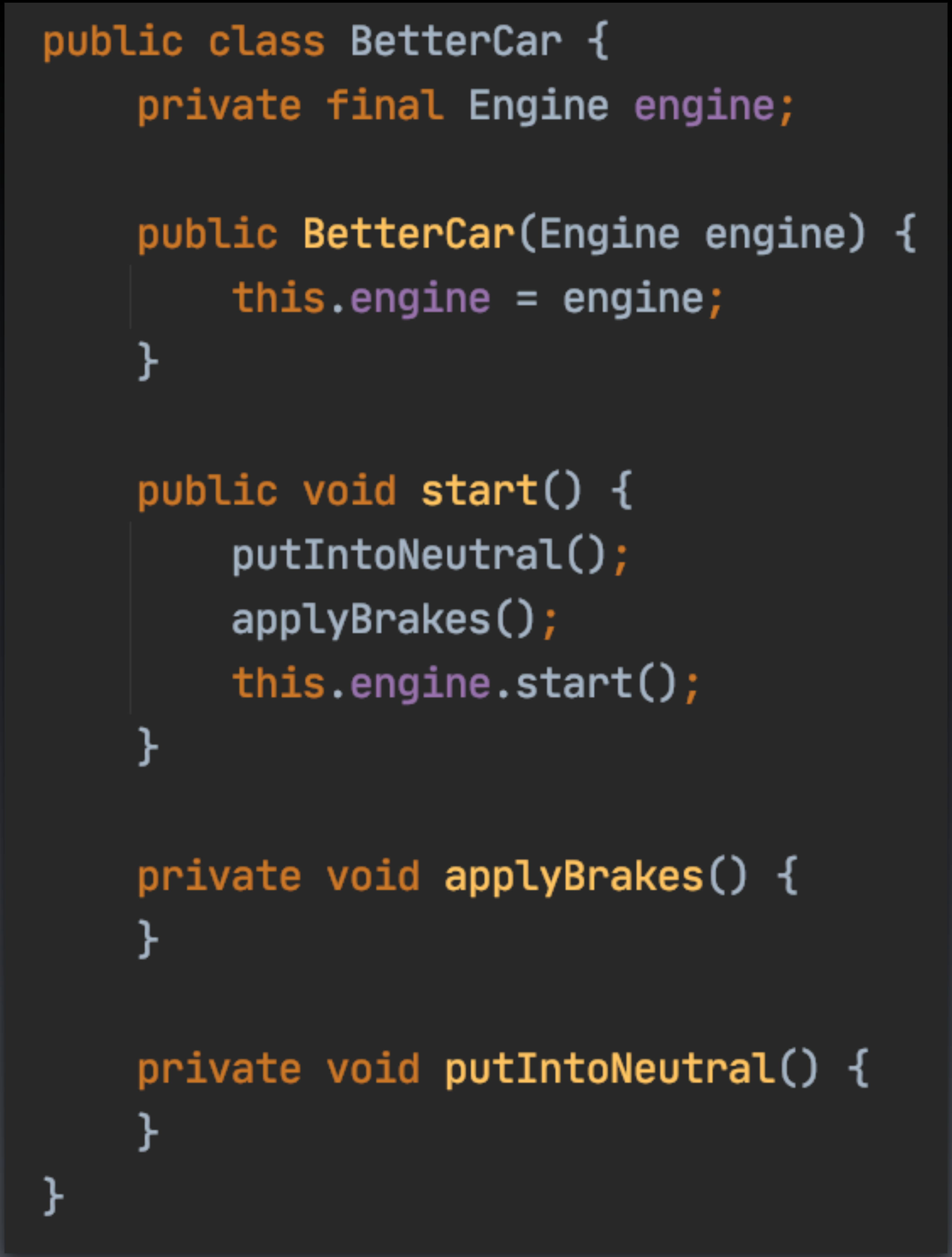


```java
public class BetterCar {
    private final Engine engine;

    public BetterCar(Engine engine) {
        this.engine = engine;
    }

    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```



```java
@Test
public void shouldStartBetterCarEngine() {
    FakeEngine engine = new FakeEngine();
    BetterCar car = new BetterCar(engine);

    car.start();

    assertTrue(engine.startedSuccessfully());
}
```

Continuous Delivery ltd



```java
public class BetterCar {
    private final Engine engine;

    public BetterCar(Engine engine) {
        this.engine = engine;
    }

    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```

Continuous Delivery ltd

```java
public class BetterCar {
    private final Engine engine;

    public BetterCar(Engine engine) {
        this.engine = engine;
    }
```

```java
public void createCars() {





}
```

```java
}
```

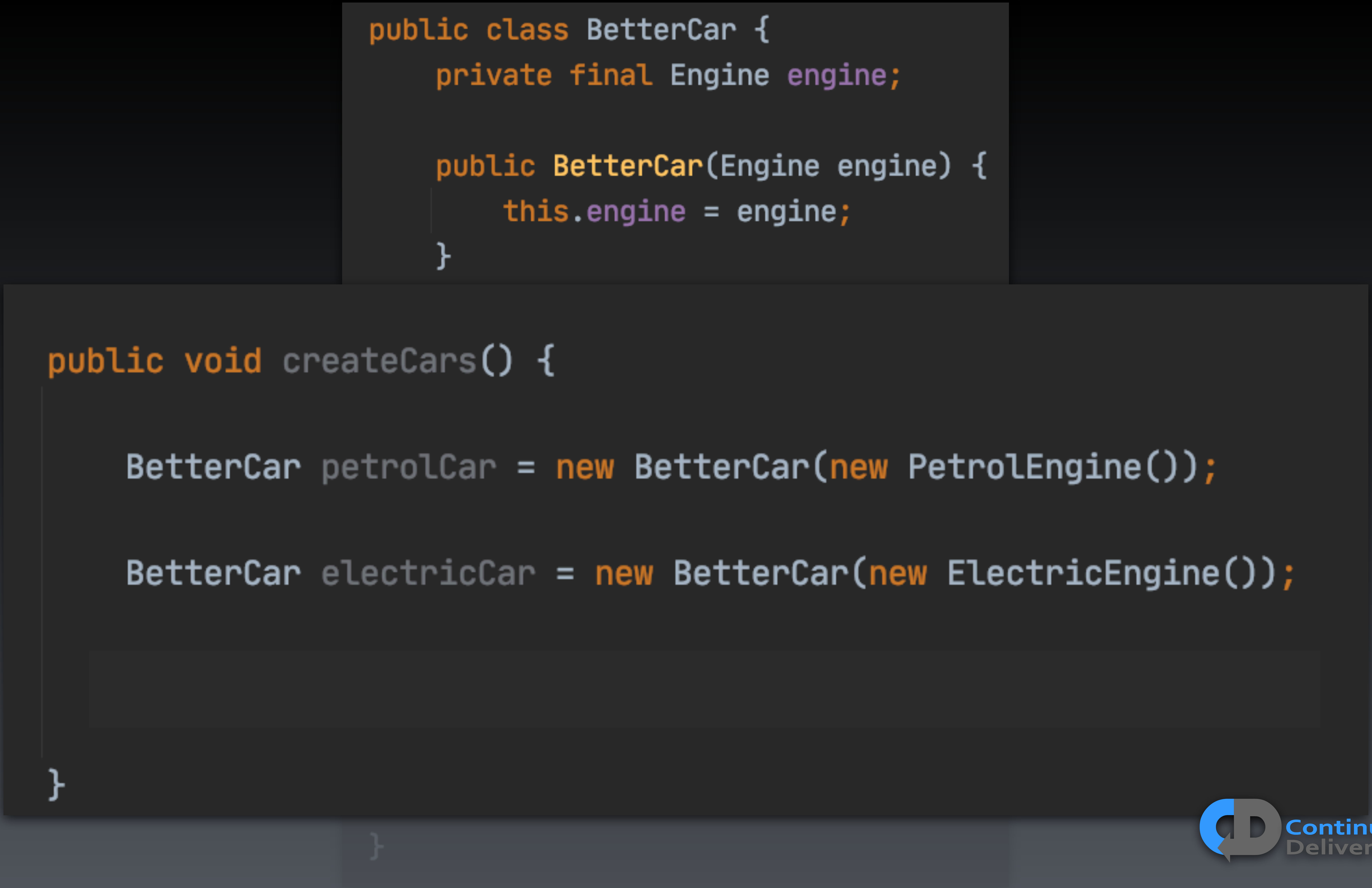
Continuous
Delivery ltd

```java
public class BetterCar {
    private final Engine engine;

    public BetterCar(Engine engine) {
        this.engine = engine;
    }
```

```java
public void createCars() {

    BetterCar petrolCar = new BetterCar(new PetrolEngine());

}
```

```java
}
```

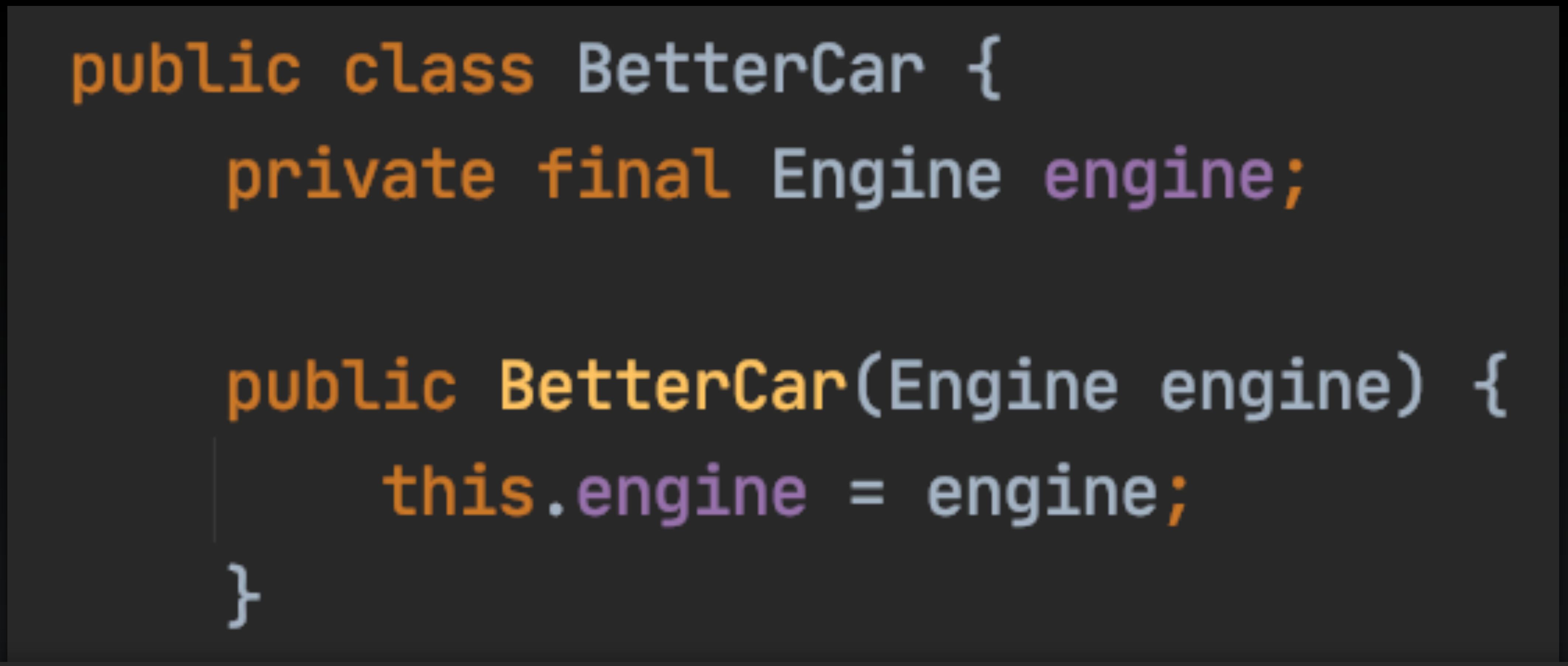
Continuous Delivery ltd



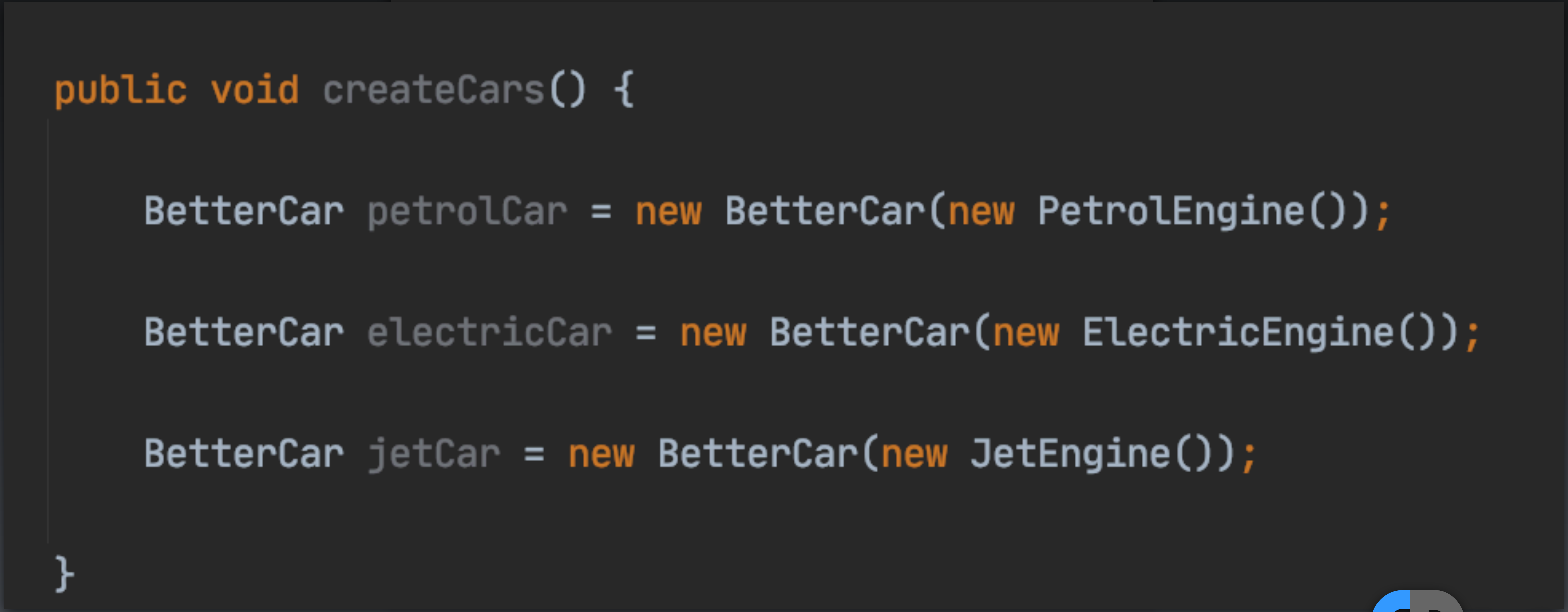
```java
public class BetterCar {
    private final Engine engine;

    public BetterCar(Engine engine) {
        this.engine = engine;
    }
```

```java
public void createCars() {

    BetterCar petrolCar = new BetterCar(new PetrolEngine());

    BetterCar electricCar = new BetterCar(new ElectricEngine());

}
```

```java
}
```



```
public class BetterCar {
    private final Engine engine;

    public BetterCar(Engine engine) {
        this.engine = engine;
    }
```



```
public void createCars() {

    BetterCar petrolCar = new BetterCar(new PetrolEngine());

    BetterCar electricCar = new BetterCar(new ElectricEngine());

    BetterCar jetCar = new BetterCar(new JetEngine());

}
```

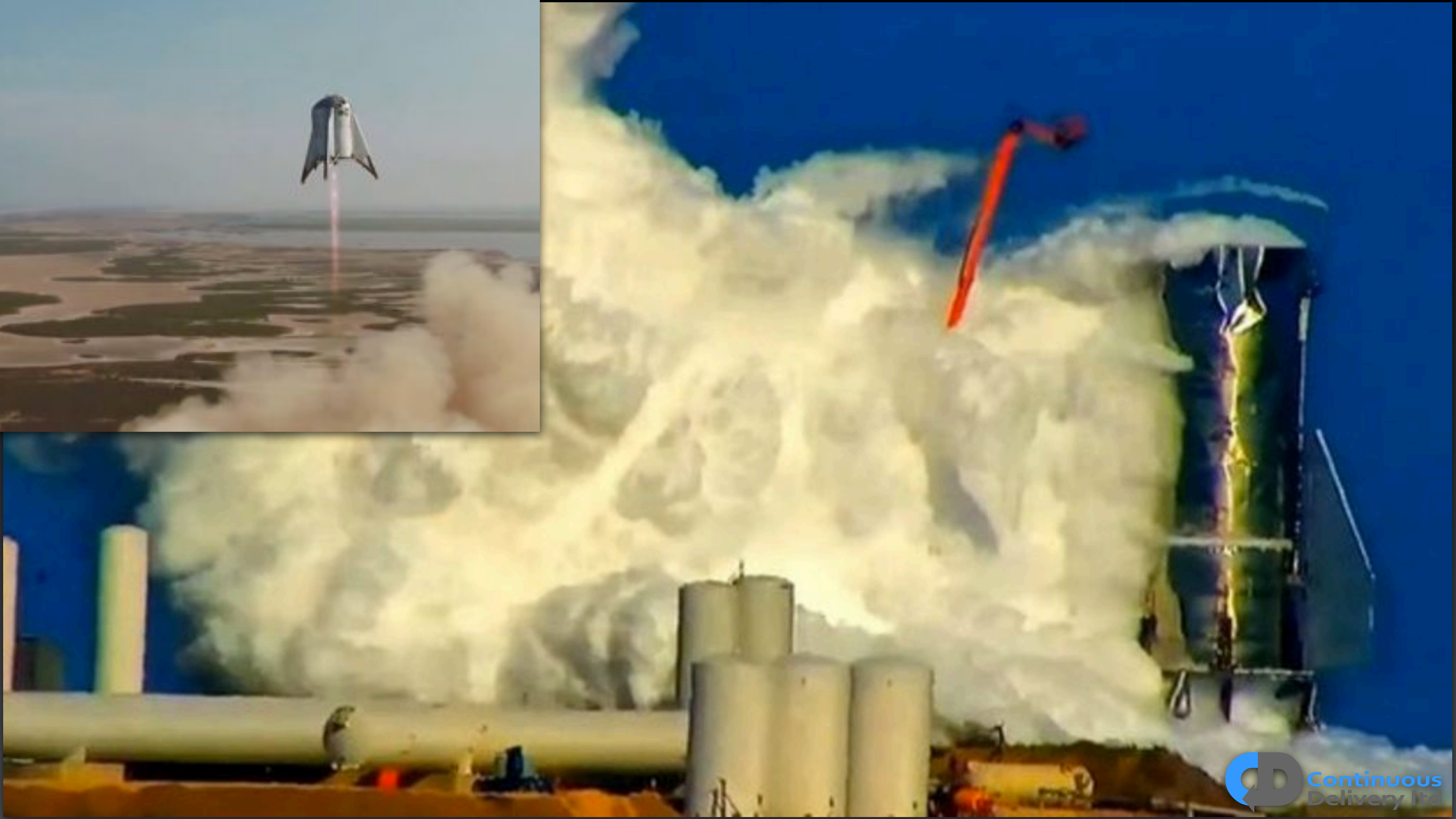
```
}
```

Working Experimentally

Continuous Delivery ltd

Continuous Delivery ltd

Continuous Delivery ltd.



Continuous Delivery ltd

Small Steps

Continuous Delivery ltd

Gather Feedback

Continuous Delivery ltd

Predict the Results

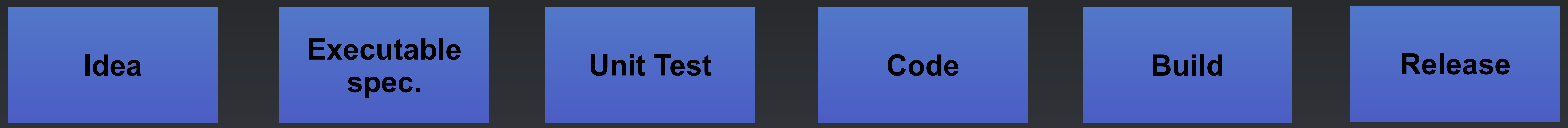
Continuous Delivery ltd

Control the Variables

Continuous Delivery ltd

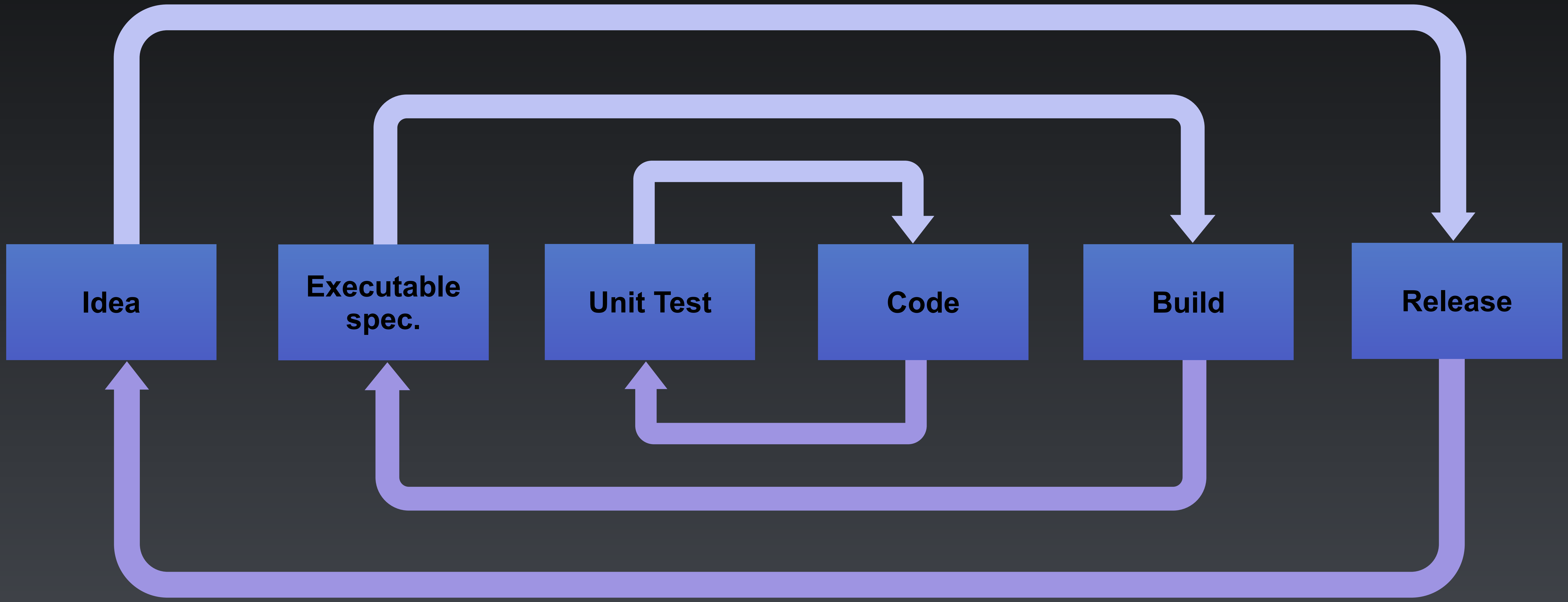
What Really Works?

Smart Automation - a repeatable, reliable process for releasing software



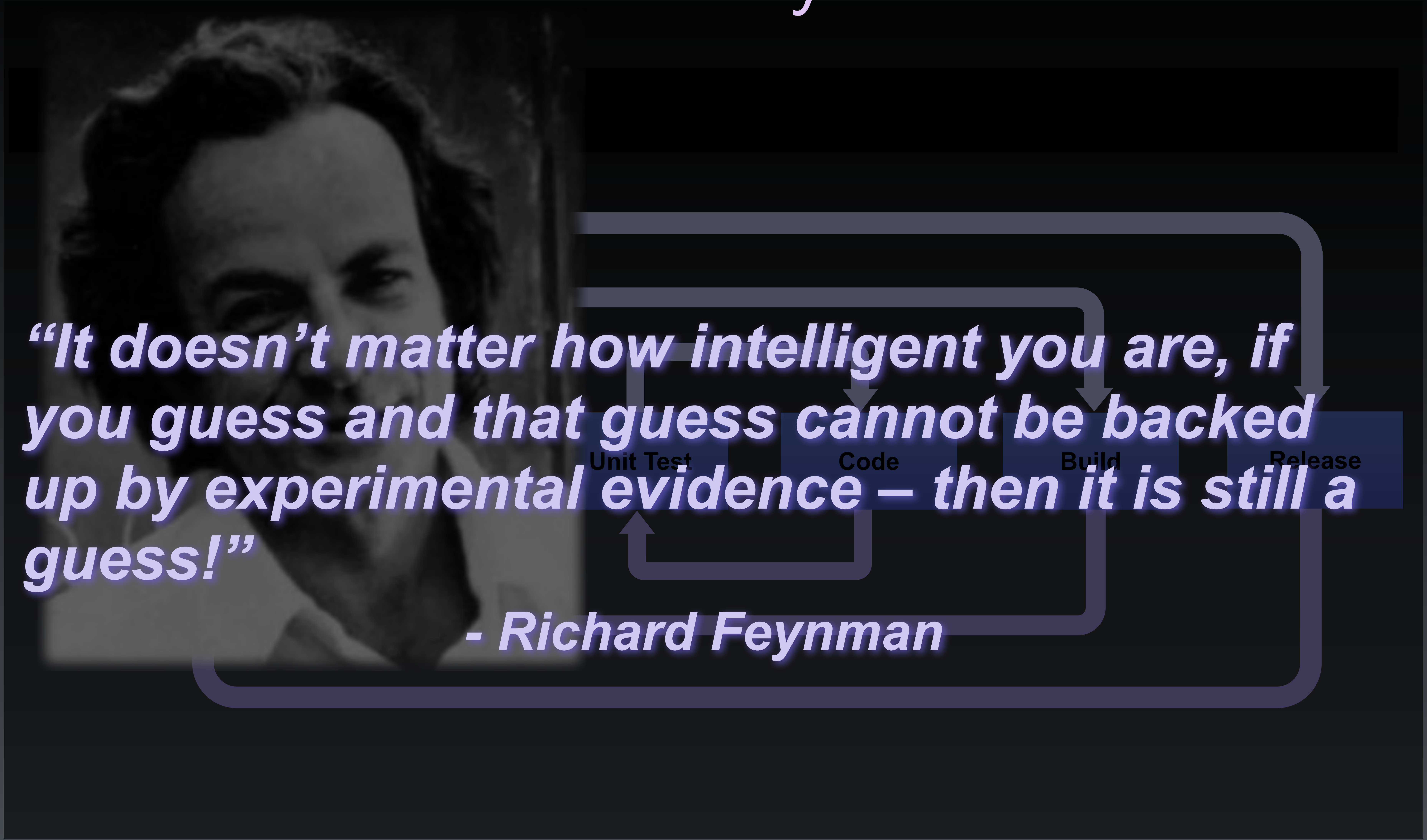
What Really Works?

Smart Automation - a repeatable, reliable process for releasing software



Continuous Delivery ltd

What Really Works?



"It doesn't matter how intelligent you are, if you guess and that guess cannot be backed up by experimental evidence – then it is still a guess!"

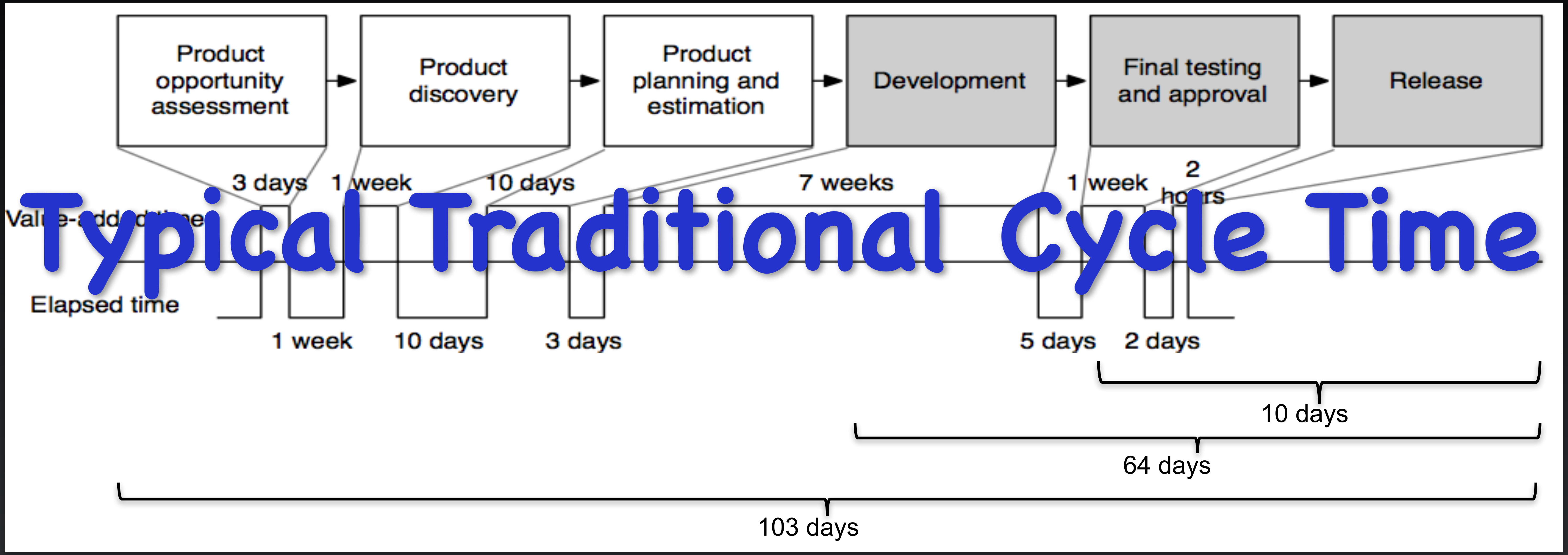
- Richard Feynman

Continuous Delivery ltd

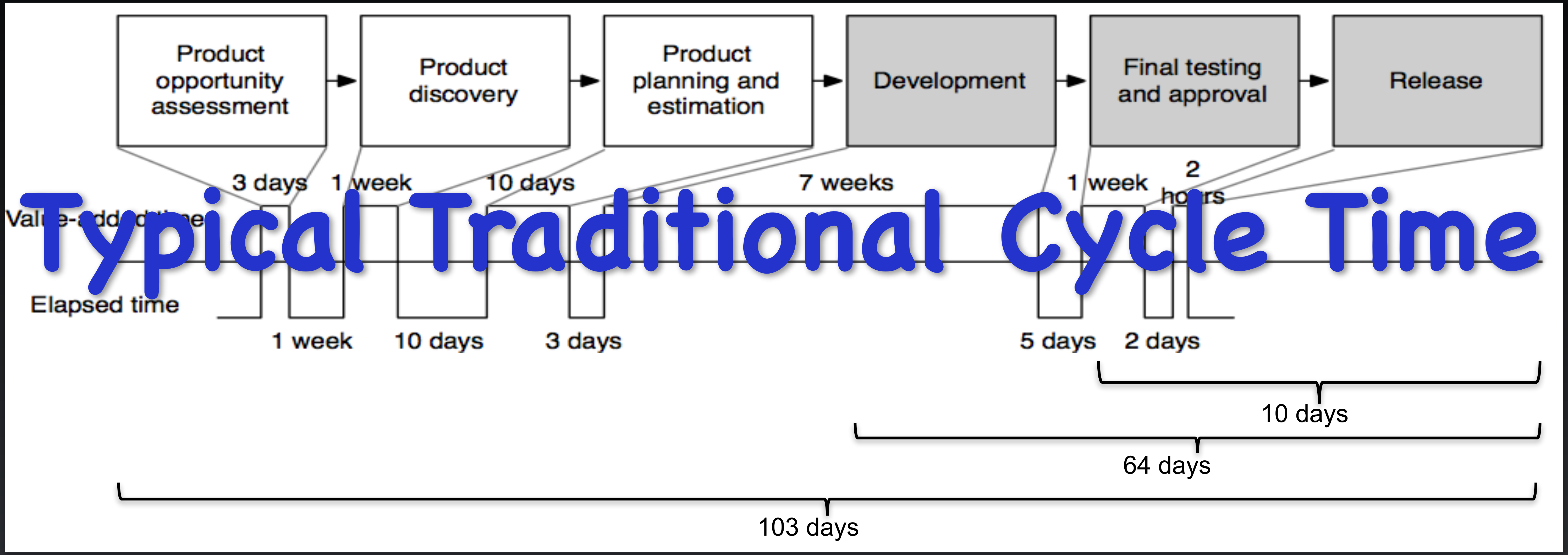
Speed

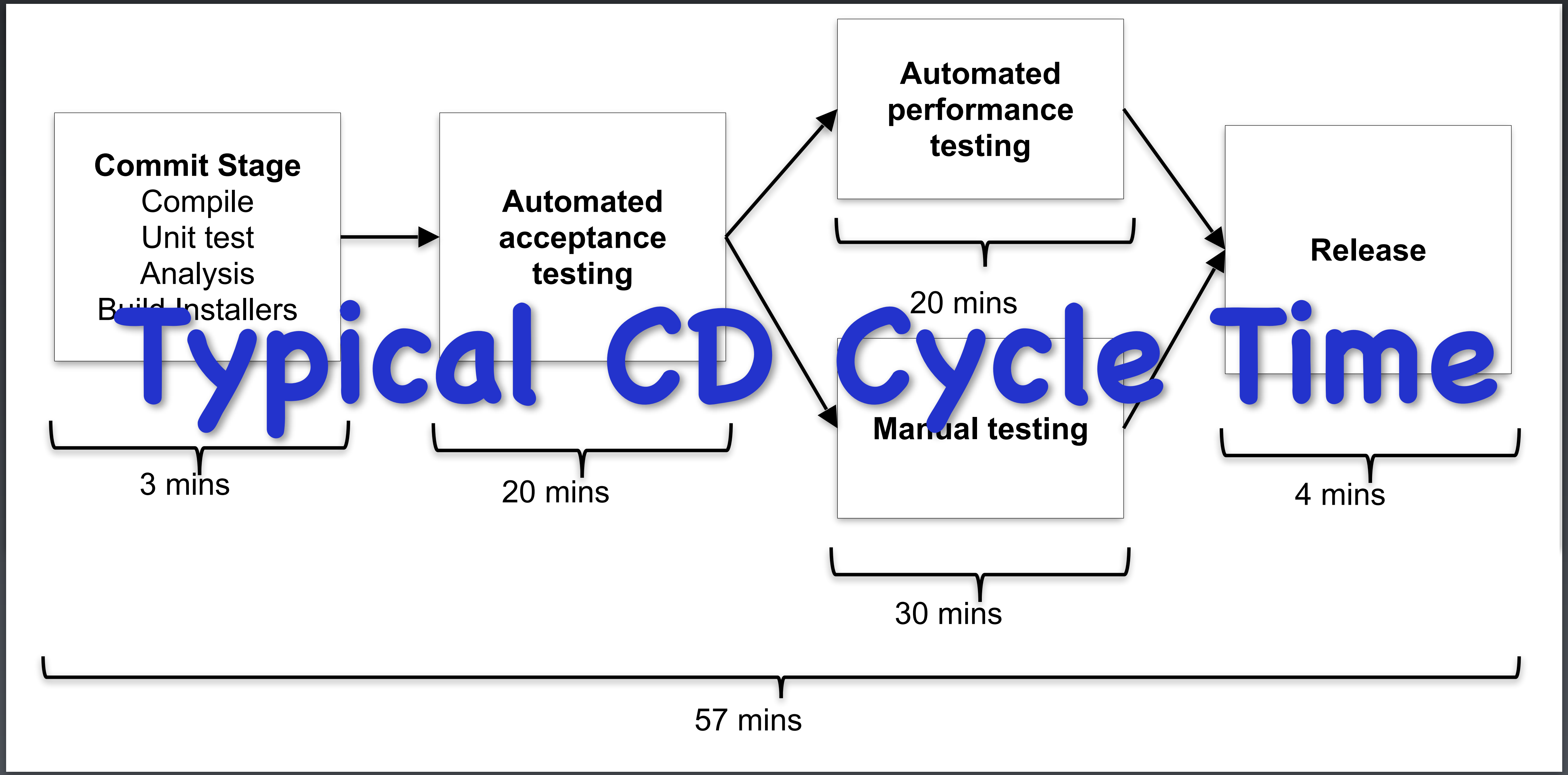
Continuous Delivery ltd

Cycle-Time


Product opportunity assessment
Product discovery
Product planning and estimation
Development
Final testing and approval
Release
3 days
1 week
10 days
7 weeks
1 week
2 hours
Value-added time
Elapsed time
1 week
10 days
3 days
5 days
2 days
10 days
64 days
103 days
Typical Traditional Cycle Time

Cycle-Time





Continuous Delivery ltd

What Is Continuous Delivery?

“Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.”

- **The first principle of the agile manifesto.**
- **The logical extension of continuous integration.**
- **A holistic approach to development.**
- **Every commit creates a release candidate.**
- **Finished means released into the hands of users, delivering value!**

The Principles of Continuous Delivery

- Create a repeatable, reliable process for releasing software.
- Automate almost everything.
- Keep everything under version control.
- If it hurts, do it more often – bring the pain forward.
- Build quality in.
- Done means released.
- Everybody is responsible for the release process.
- Improve continuously.

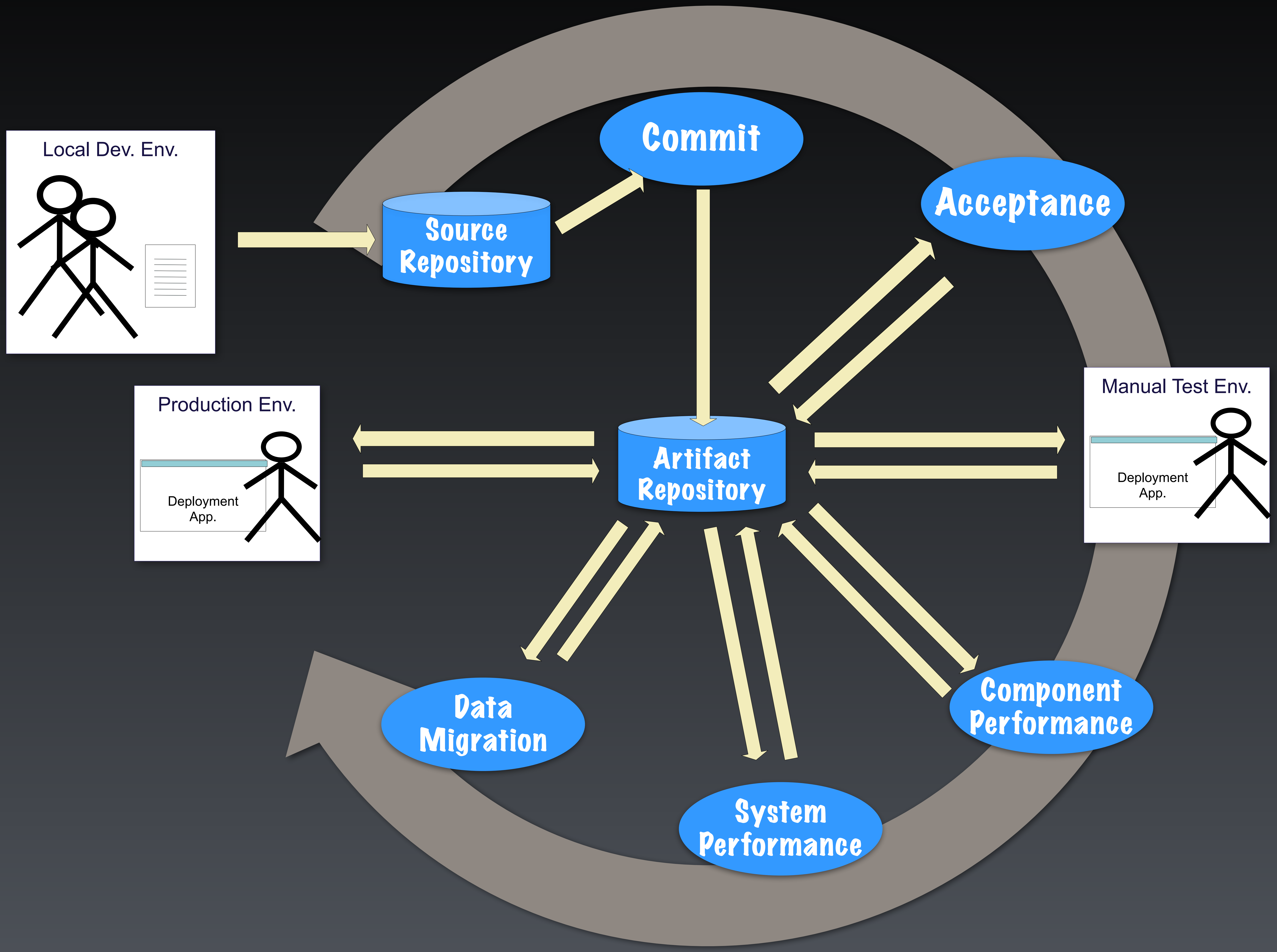
The Principles of Continuous Delivery

- Create a repeatable, reliable process for releasing software.
- Automate almost everything.
- Keep everything under version control.
- If it hurts, do it more often, and bring the pain forward.
- Build quality in.
- Done means released.
- Everybody is responsible for the release process.
- Improve continuously.

“If Agile software development was the opening act to a great performance, Continuous Delivery is the headliner.”

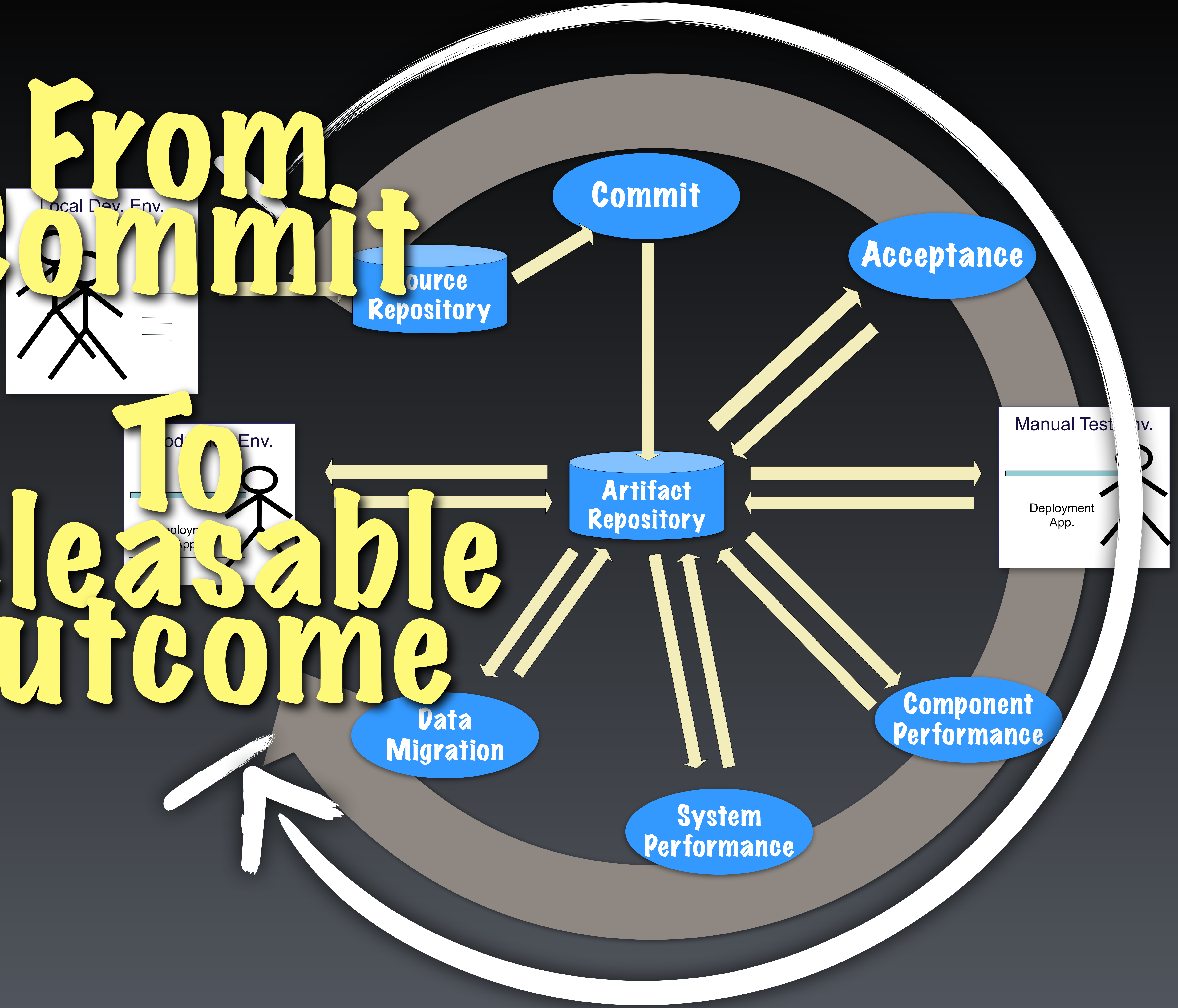
Forrester Research 2013

Continuous Delivery ltd

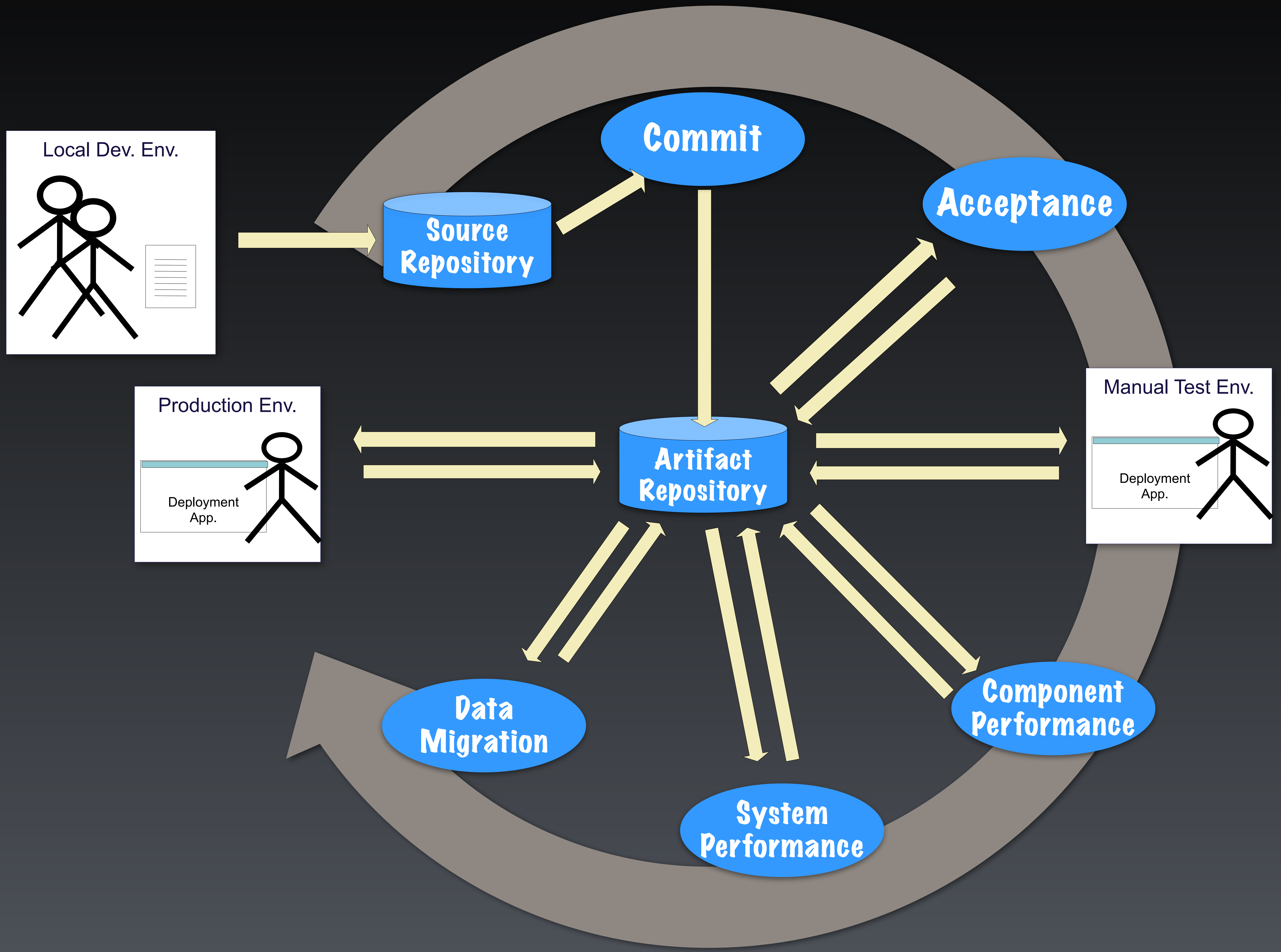


Continuous Delivery ltd

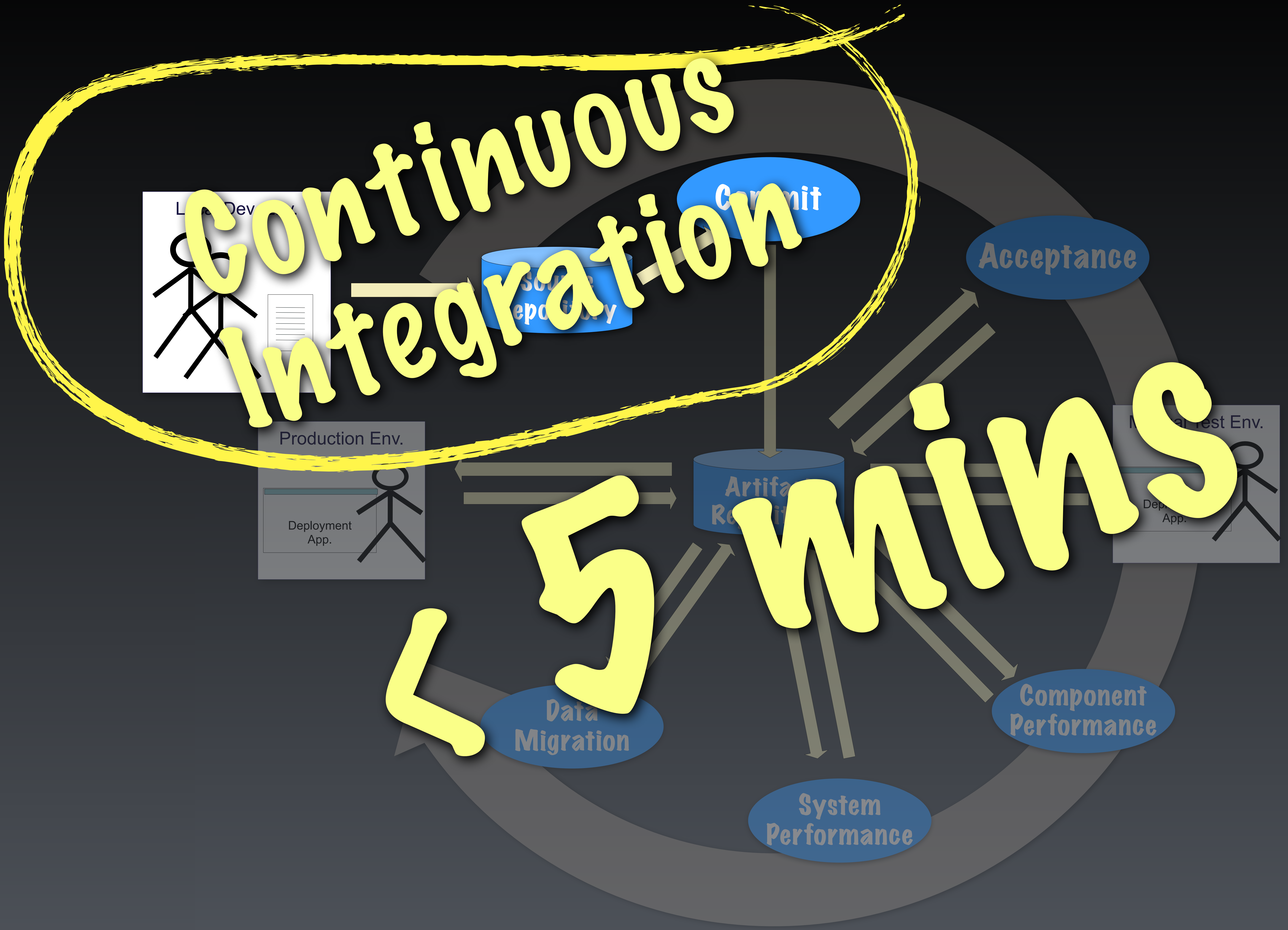
From Commit To Releasable Outcome



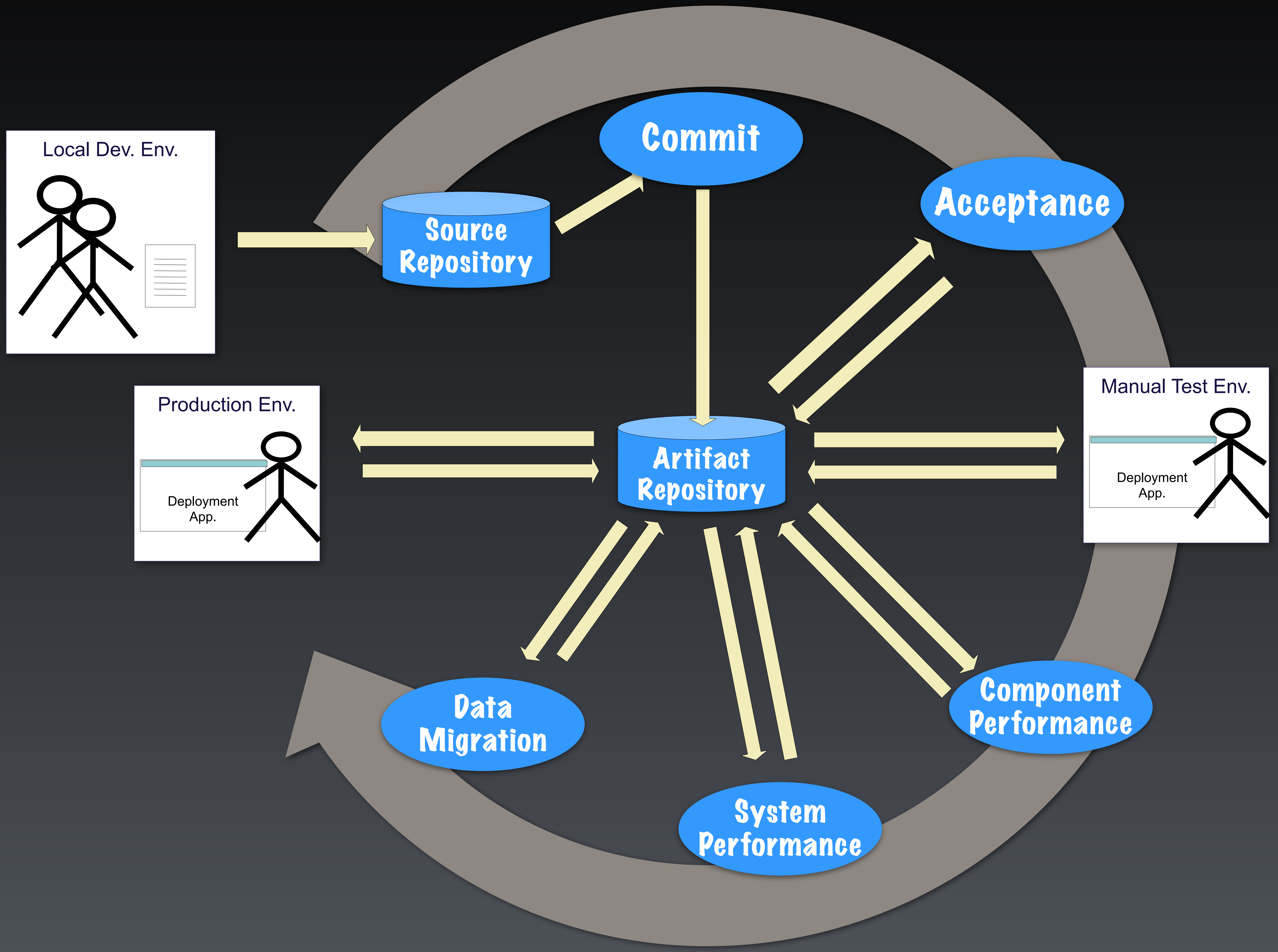
Continuous Delivery ltd



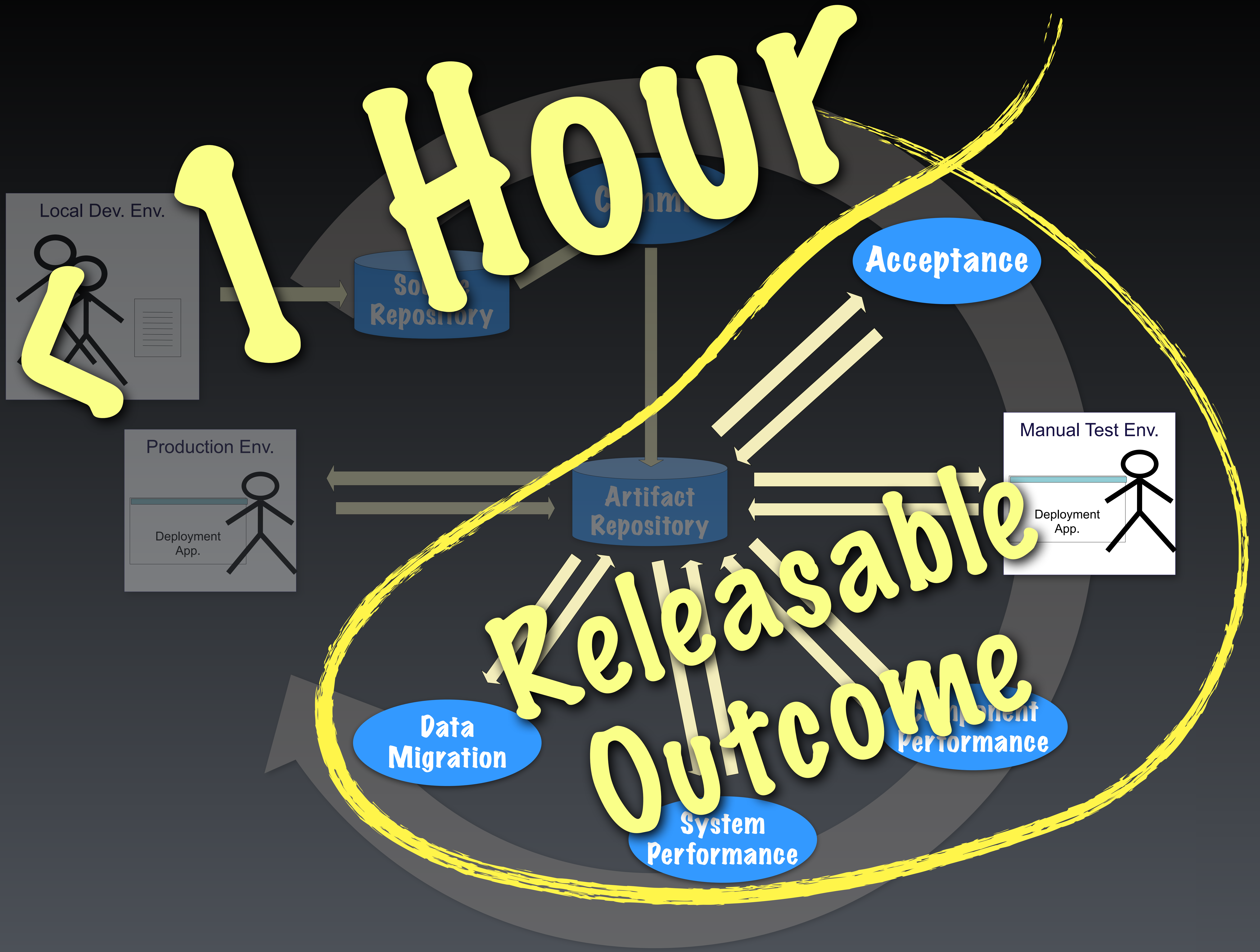
Continuous Delivery ltd



Continuous Delivery ltd



Continuous Delivery ltd

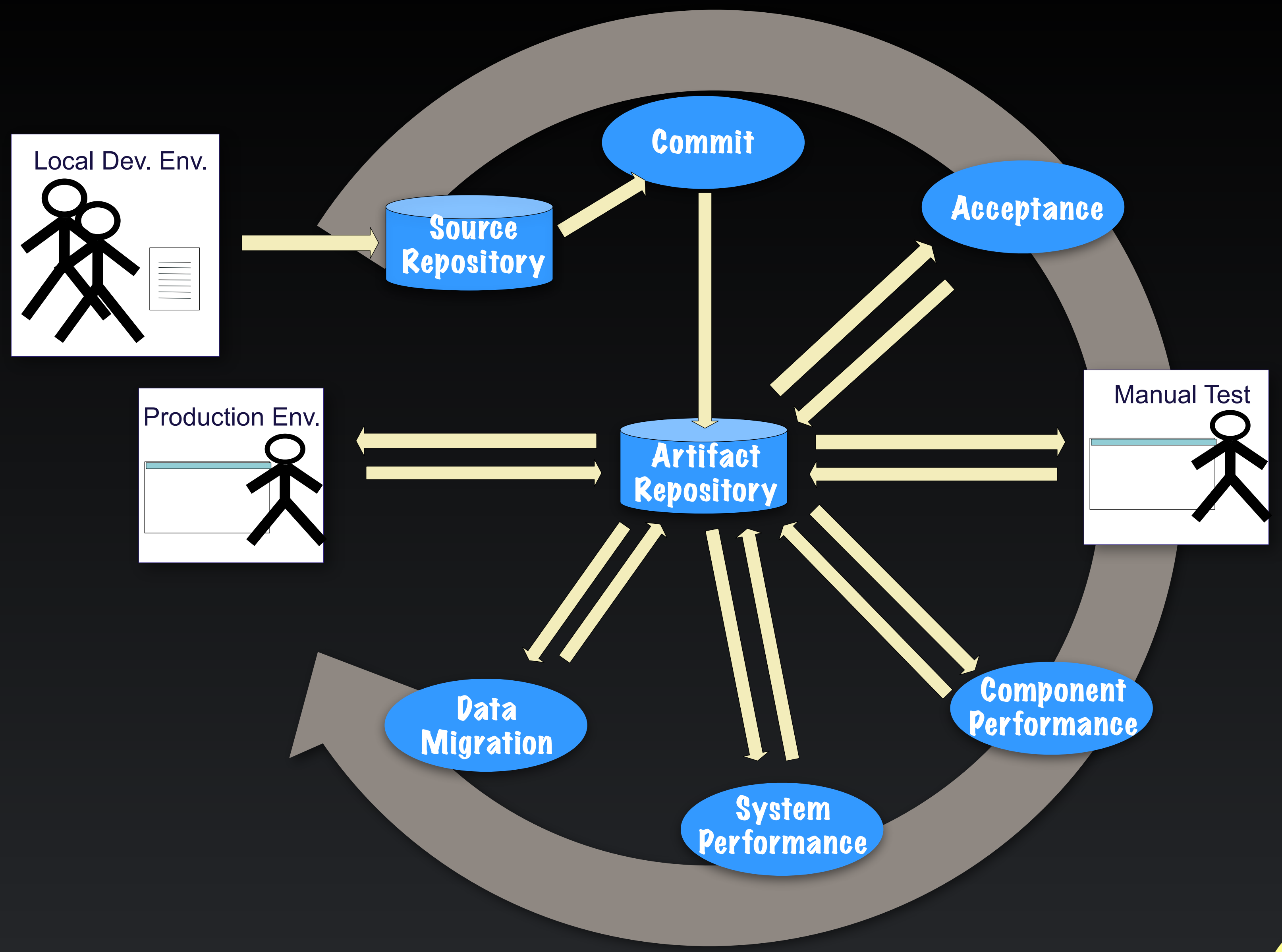


Continuous Delivery ltd

Engineering: Design, Simulate, Build

Alan Kay

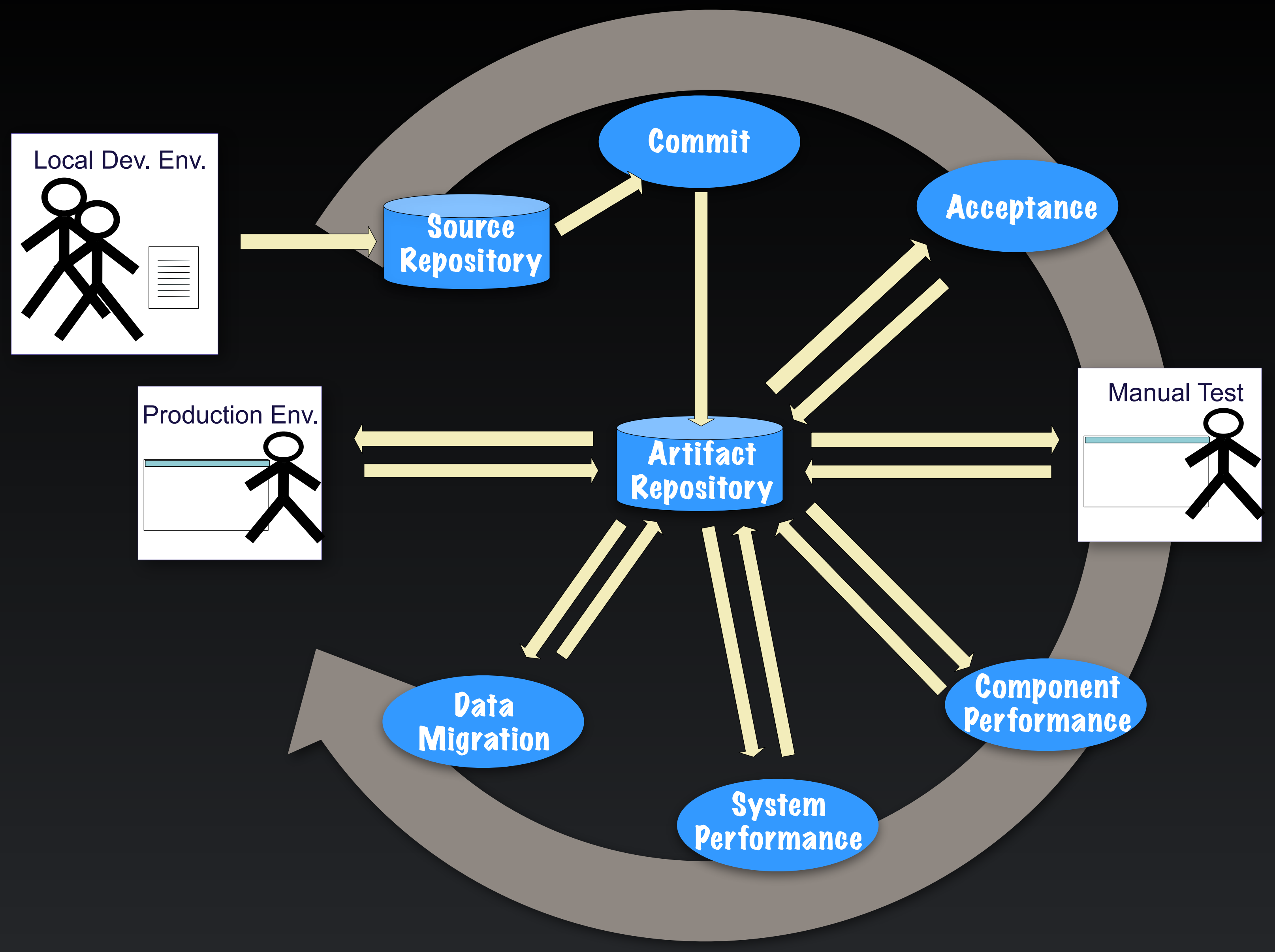
Continuous Delivery ltd



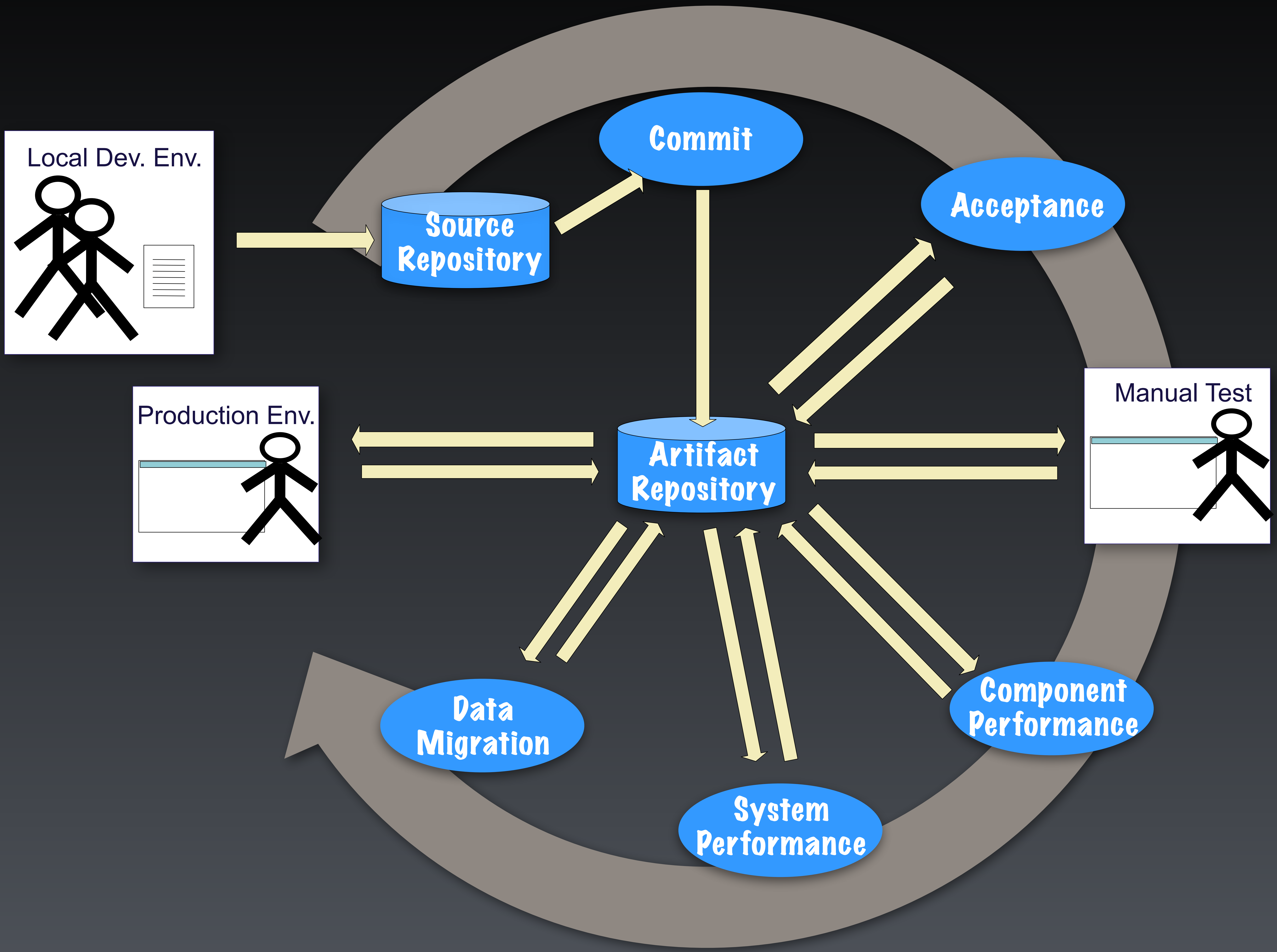
Engineering: Design, Simulate, Build

Alan Kay

Continuous Delivery ltd

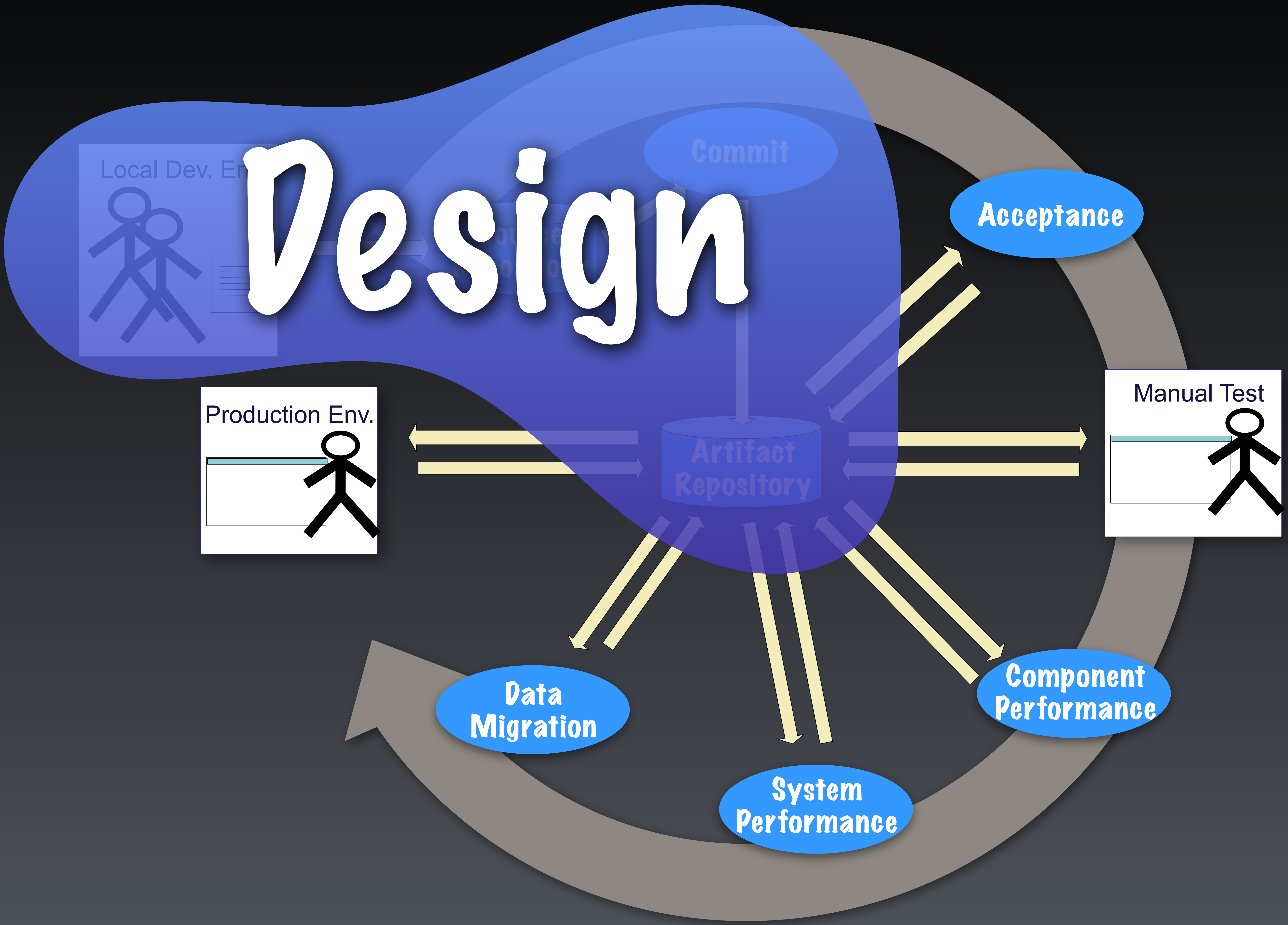


Continuous Delivery ltd

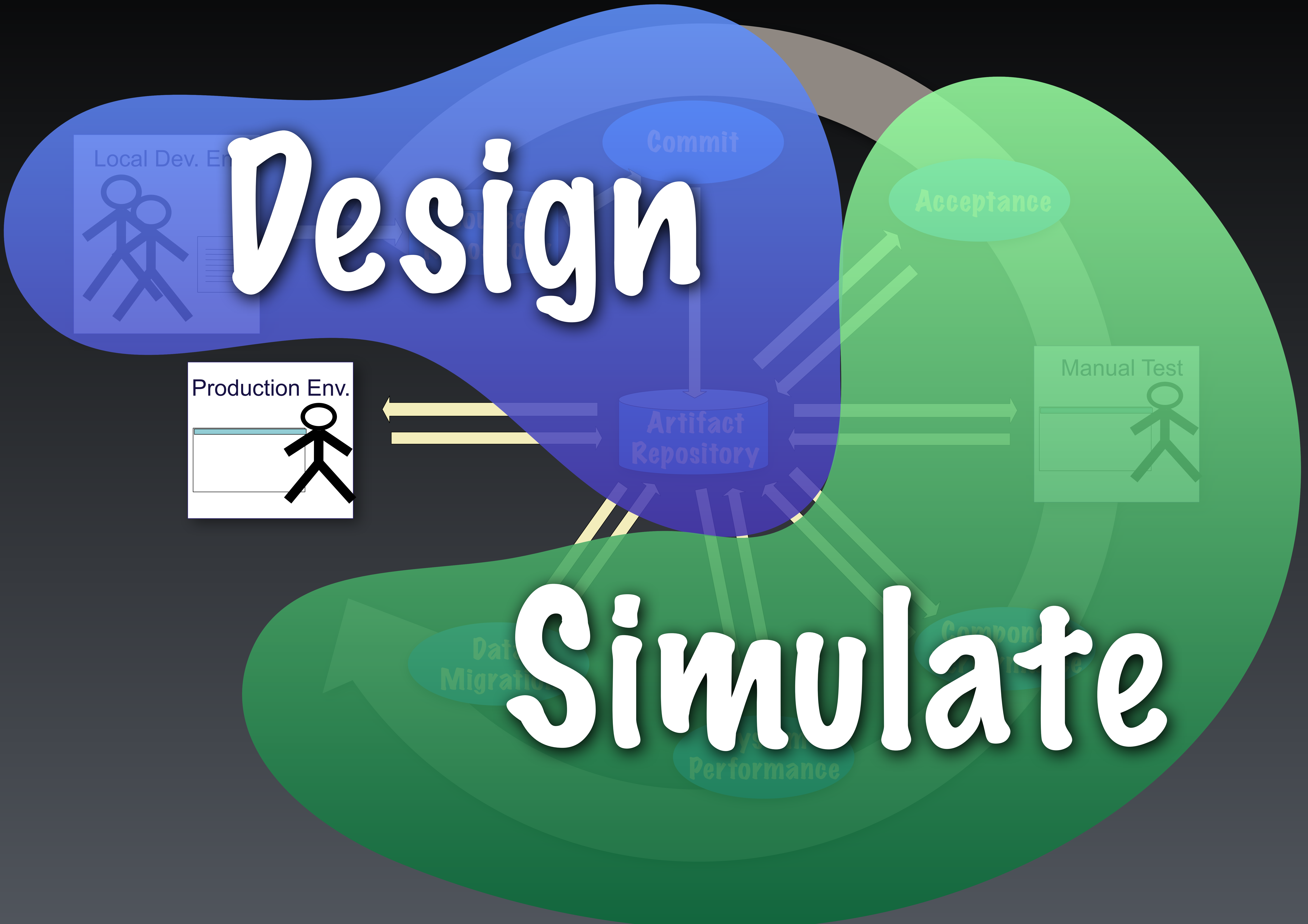


Continuous Delivery ltd

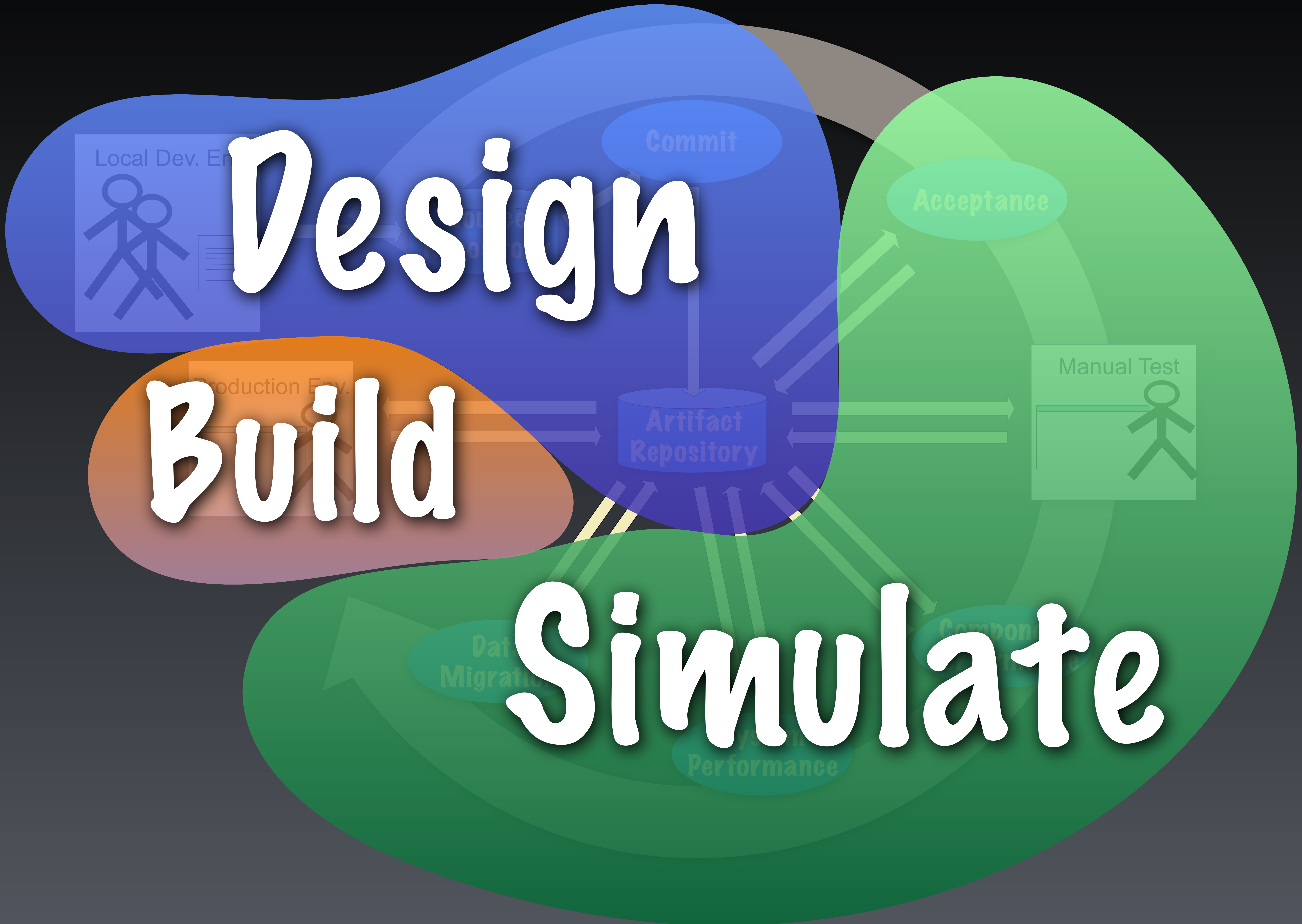
Design



Continuous Delivery ltd



Continuous Delivery ltd

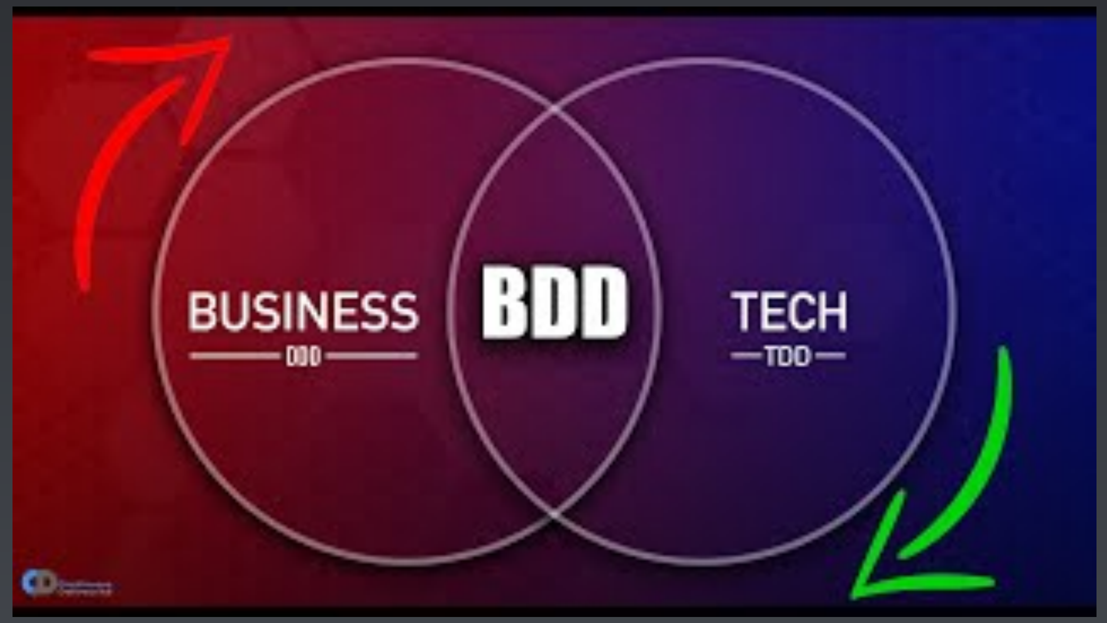


Continuous Delivery ltd

https://bit.ly/CDonYT

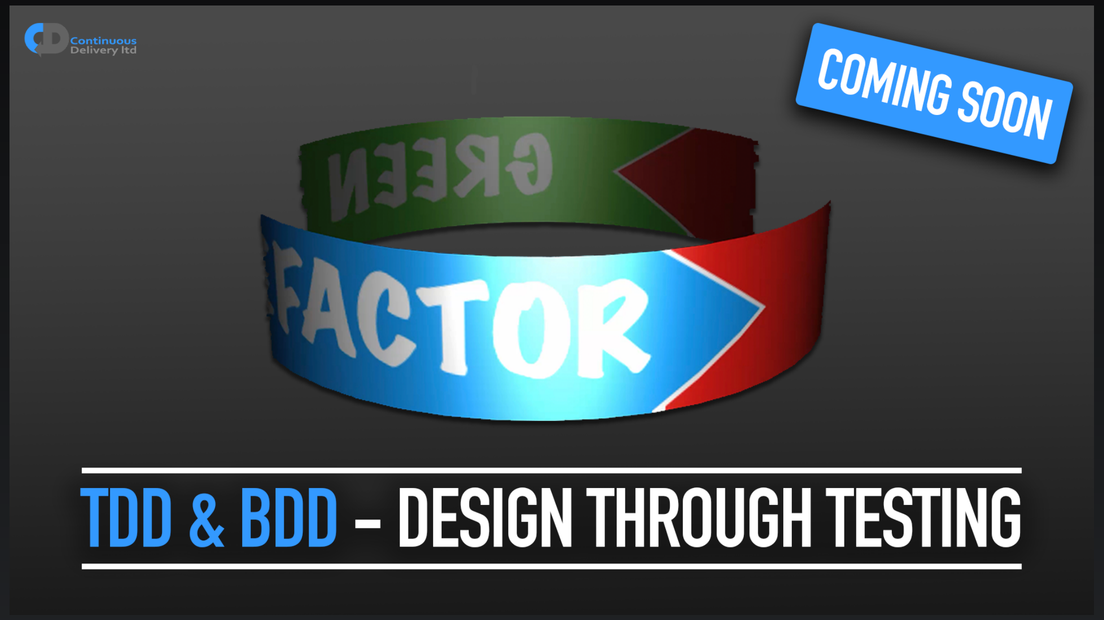
Continuous Delivery ltd

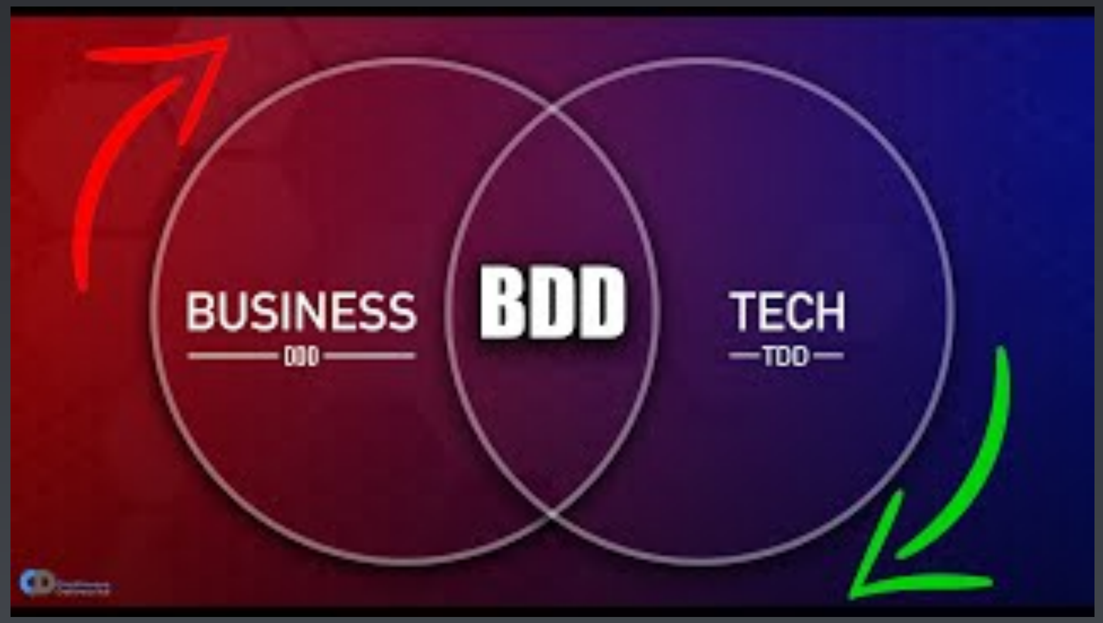
https://bit.ly/CDonYT



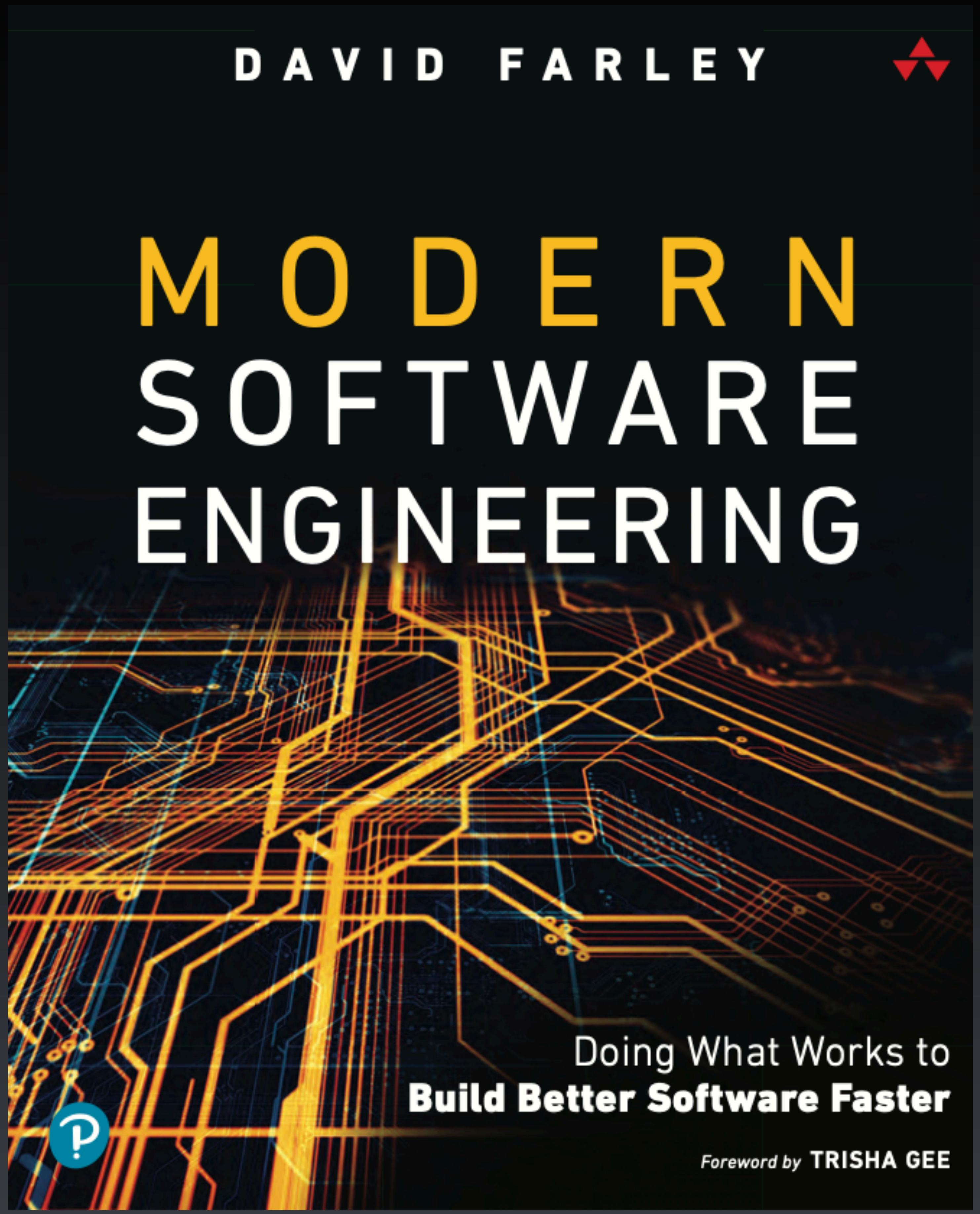
Continuous Delivery ltd

https://bit.ly/CDonYT





Continuous Delivery ltd

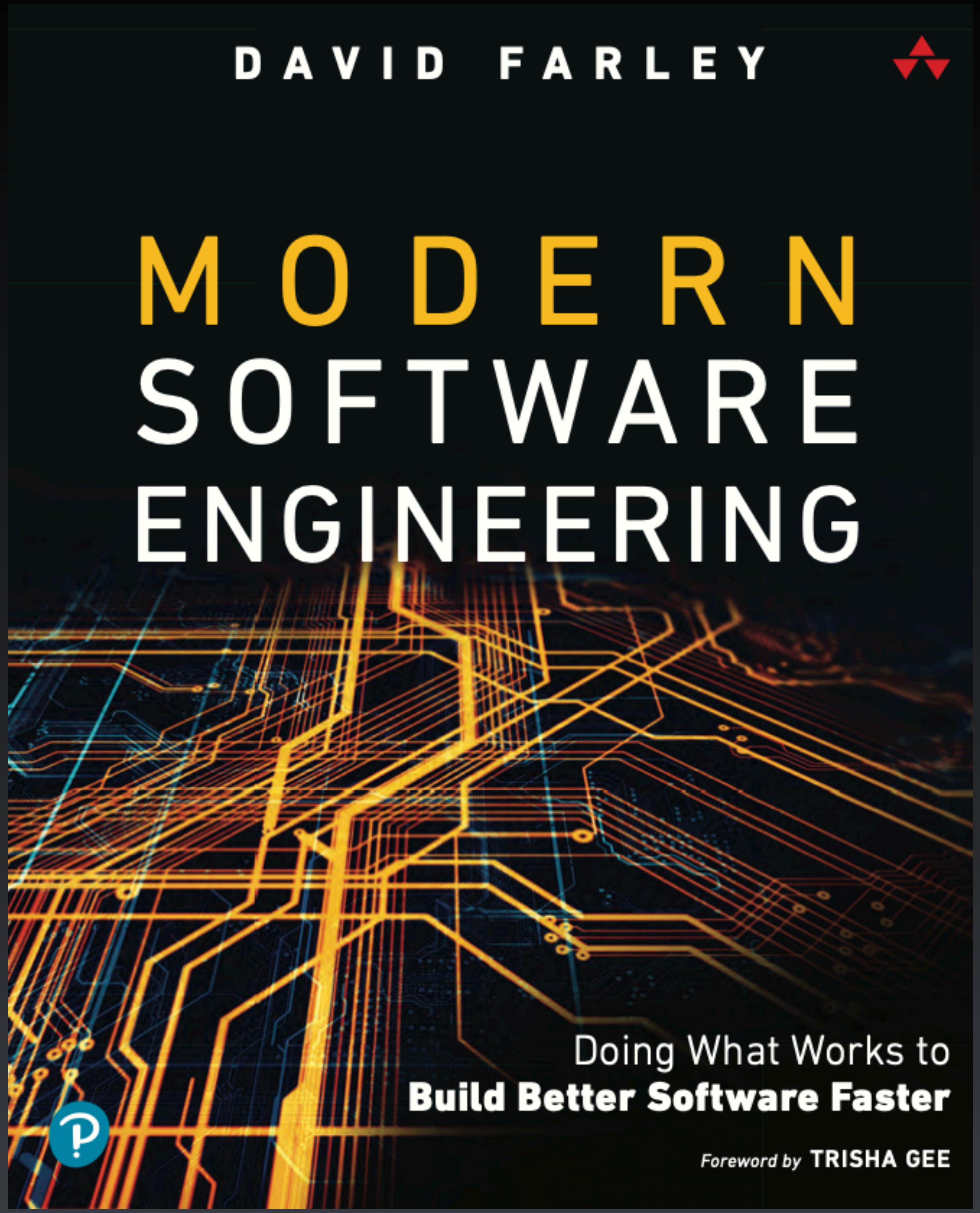

DAVID FARLEY

MODERN SOFTWARE ENGINEERING

Doing What Works to
Build Better Software Faster

Foreword by TRISHA GEE

Continuous Delivery ltd



Use the Code:

MODERNSWENG

For 35% Off!

Pre-Order

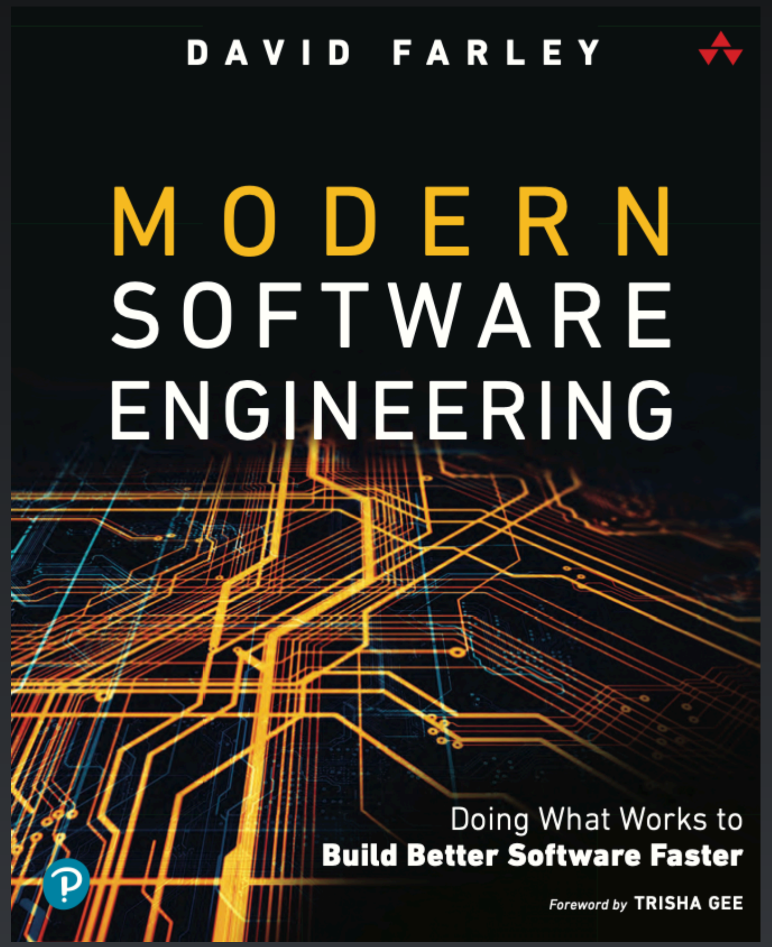
➡️ https://informit.com/modernsweng

Continuous Delivery ltd

Q&A

Continuous Delivery ltd

http://www.continuous-delivery.co.uk

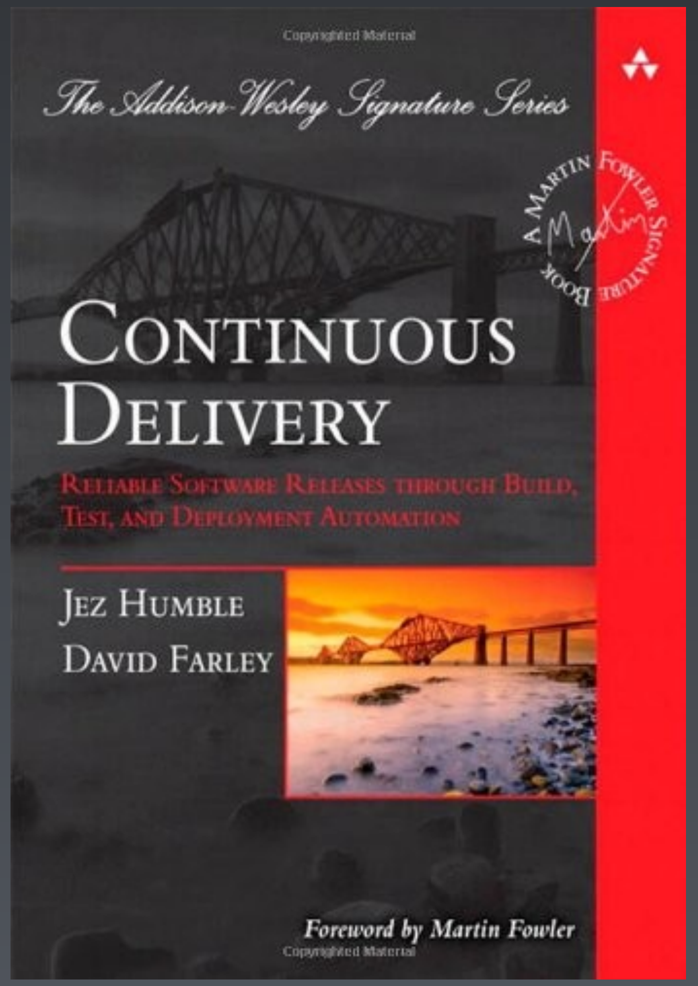

DAVID FARLEY
MODERN SOFTWARE ENGINEERING
Doing What Works to Build Better Software Faster
Foreword by TRISHA GEE

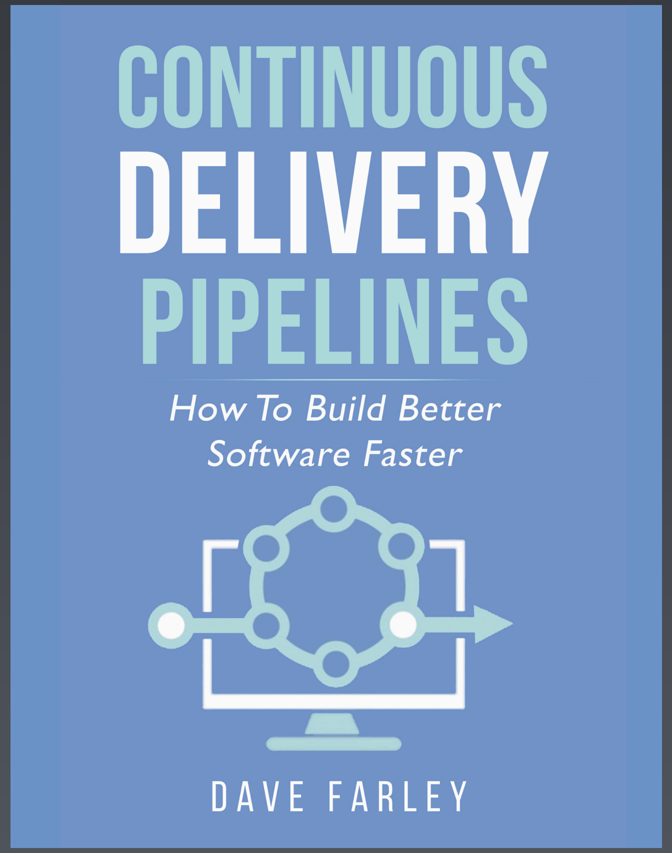
Dave Farley

https://www.davefarley.net

@davefarley77

https://bit.ly/CDonYT


The Addison-Wesley Signature Series
CONTINUOUS DELIVERY
Reliable Software Releases through Build, Test, and Deployment Automation
Jez Humble
David Farley
Foreword by Martin Fowler


CONTINUOUS DELIVERY PIPELINES
How To Build Better Software Faster
DAVE FARLEY