Null Check Analysis in Java Code

Manuel Leuenberger

Software Composition Seminar

05.05.2015

Motivation

- NullPointerException-related bugs are the most frequent in Java projects
- Often fixed by guarding problematic code with a null check

- What are the *things* that are checked against null?
- Where are these *things* coming from?

```
public class CoDriver {
  …
  public void join(Driver driver) {
    …
    Car car;
    …
    car = driver.getCar();
    …
    if (car != null) {
    …
  }
}
```

```
public class CoDriver {
   …
   public void join(Driver driver) {
      …
      Car car;
      …
      car = driver.getCar();
      …
      if (car != null) {
      …
   }
}
```

```
public class CoDriver {
  …
  public void join(Driver driver) {
    …
    Car car;
    …
    car = driver.getCar();
    …
    if (car != null) {
    …
  }
}
```

```
public class CoDriver {
  …
  public void join(Driver driver) {
    …
    Car car;
    …
    car = driver.getCar();
    …
    if (car != null) {
    …
  }
}
```

We call this *thing*
comparand

```
public class CoDriver {
  …
  public void join(Driver driver) {
    …
    Car car;
    …
    car = driver.getCar();
    …
    if (car != null) {
    …
  }
}
```

name

```
public class CoDriver {
  …
  public void join(Driver driver) {
    …
    Car car;
    …
    car = driver.getCar();
    …
    if (car != null) {
    …
  }
}
```

```
public class CoDriver {
  …
  public void join(Driver driver) {
    …
    Car car;
    …
    car = driver.getCar();
    …
    if (car != null) {
    …
  }
}
```

```
public class CoDriver {
   …
   public void join(Driver driver) {
      …
      Car car;
      …
      car = driver.getCar();
      …
      if (car != null) {
      …
   }
}
```

local variable

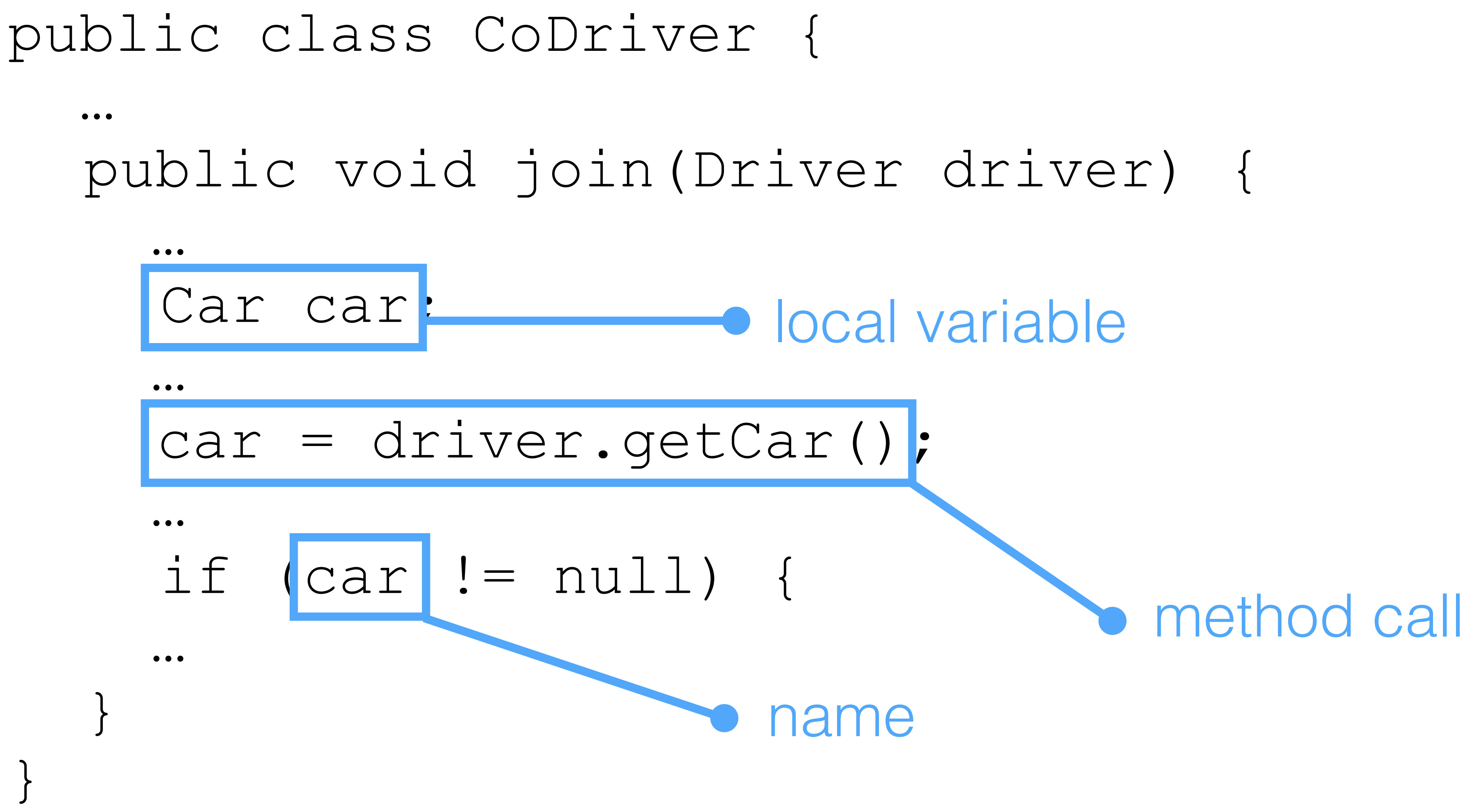
```
public class CoDriver {
   …
   public void join(Driver driver) {
      …
      Car car;
      …
      car = driver.getCar();
      …
      if (car != null) {
      …
   }
}
```

```
public class CoDriver {
  …
  public void join(Driver driver) {
    …
    Car car;
    …
    car = driver.getCar();
    …
    if (car != null) {
    …
  }
}
```

```
public class CoDriver {
  …
  public void join(Driver driver) {
    …
    Car car;
    …
    car = driver.getCar();
    …
    if (car != null) {
    …
  }
}
```

method call

```
public class CoDriver {
  …
  public void join(Driver driver) {
    …
    Car car;
    …
    car = driver.getCar();
    …
    if (car != null) {
    …
  }
}
```



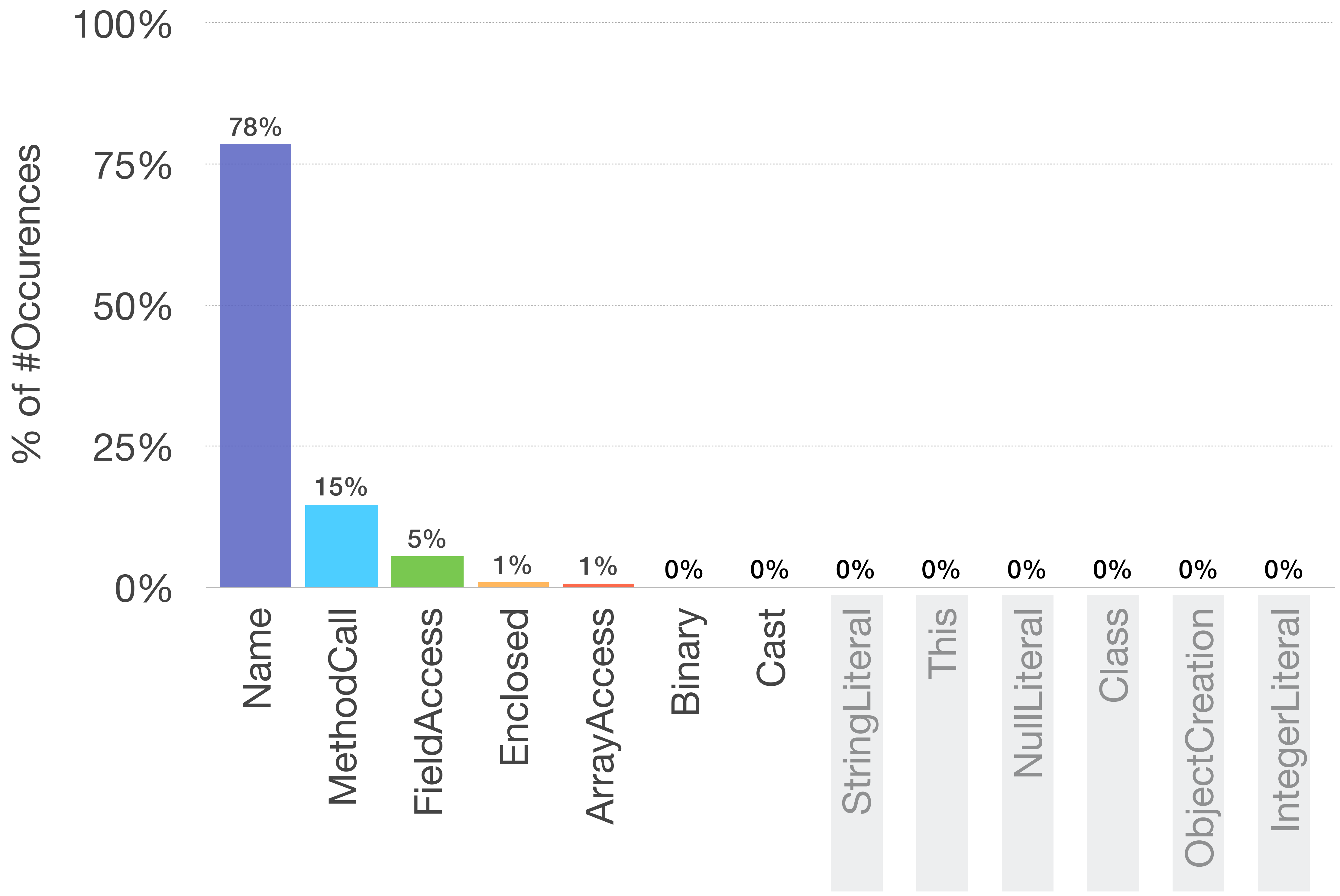
```
public class CoDriver {
  …
  public void join(Driver driver) {
    …
    Car car;
    …
    car = driver.getCar();
    …
    if (car != null) {
    …
  }
}
```

Dataset

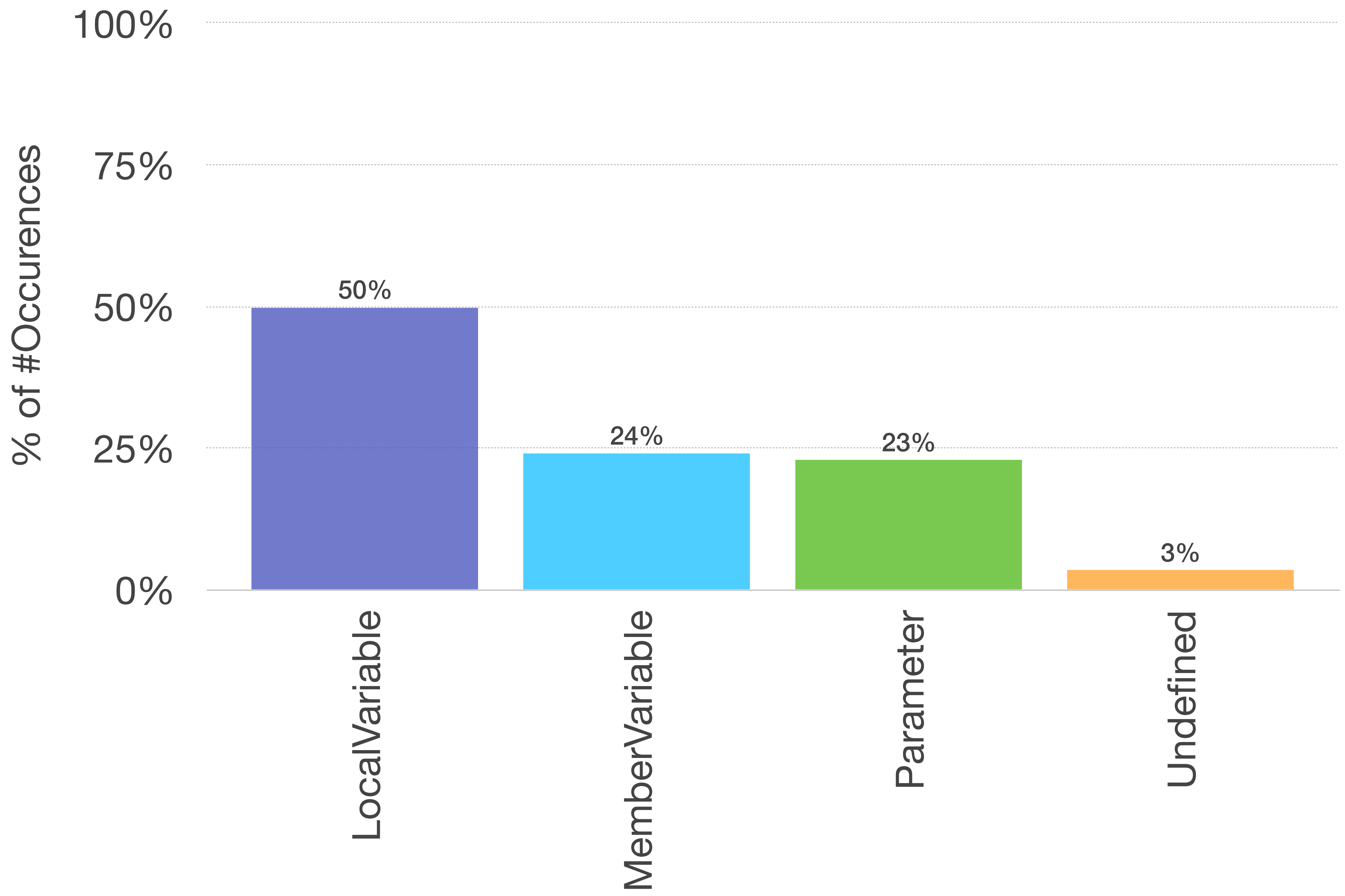
- 717 projects
- 374’203 Java sources processed
- 6’106 Java sources unprocessable

35% of conditionals are null checks!

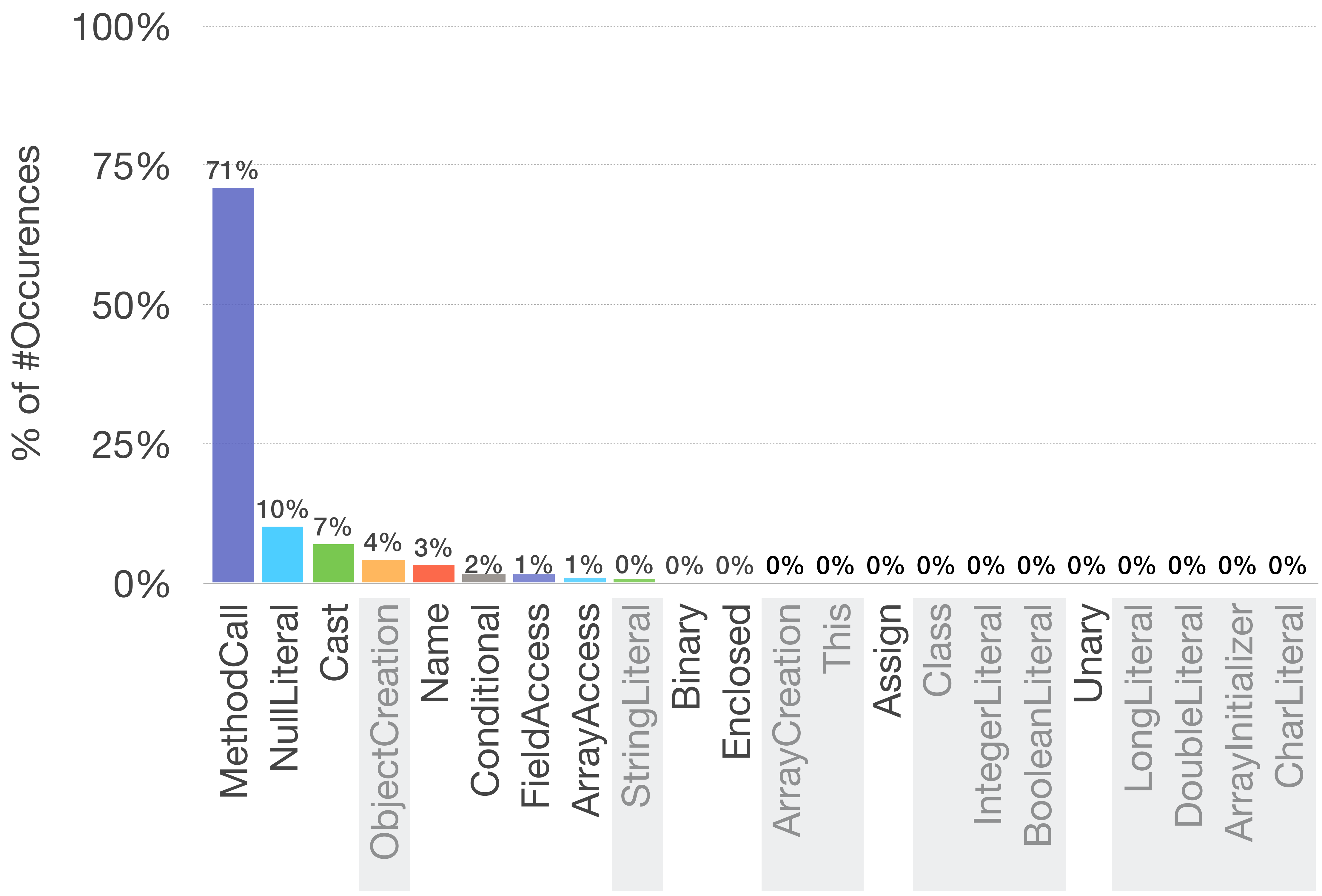
Comparands are…



Assignable comparands are…



Assignable comparands come from…



Conclusion

- We check for null, because methods return null
- …but we often mistakenly do not expect null

Lessons Learned

- Spring B tch: Too big to handle

- Performance: Caching is your friend

- MySQL: Question your very basic mental model

+ Java 8 Streams: Process collections like a boss

+ GraphViz: Got graph data? Got visualization!