Evaluating tests

Criteria

- Coverage
  - How much code is tested?
- Performance
  - How fast can tests be run?

Coverage

$$\frac{Code\ executed\ by\ tests}{Total\ code}$$

“Code”?

- Lines?
- Statements?
- Branches?
- …?

Coverage

```
int getPriority(User user) {
  if (user == null) throw ...;
  int priority = 100;
  if (user.isInfluential()) priority += 100;
  if (user.hasUnpaidBills()) priority /= 2;
  return priority;
}
```

Line coverage?

Influential?	Unpaid bills?
True	True

```
int getPriority(User user) {
  if (user == null) throw ...;
  int priority = 100;
  if (user.isInfluential()) priority += 100;
  if (user.hasUnpaidBills()) priority /= 2;
  return priority;
}
```

Line coverage?

```
int getPriority(User user) {
  if (user == null) throw ...;
  int priority = 100;
  if (user.isInfluential()) priority += 100;
  if (user.hasUnpaidBills()) priority /= 2;
  return priority;
}
```

Statement coverage?

Influential?	Unpaid bills?
True	True

```
int getPriority(User user) {
  if (user == null) throw ...;
  int priority = 100;
  if (user.isInfluential()) priority += 100;
  if (user.hasUnpaidBills()) priority /= 2;
  return priority;
}
```

Statement coverage?

```
int getPriority(User user) {
  if (user == null) throw ...;
  int priority = 100;
  if (user.isInfluential()) priority += 100;
  if (user.hasUnpaidBills()) priority /= 2;
  return priority;
}
```

Branch coverage?

Influential?	Unpaid bills?
True	True

```
int getPriority(User user) {
  if (user == null) throw ...;
  int priority = 100;
  if (user.isInfluential()) priority += 100;
  if (user.hasUnpaidBills()) priority /= 2;
  return priority;
}
```

Branch coverage?

Influential?	Unpaid bills?
True	True

```
int getPriority(User user) {
  if (user == null) throw ...;
  int priority = 100;
  if (user.isInfluential()) priority += 100;
  if (user.hasUnpaidBills()) priority /= 2;
  return priority;
}
```

Branch coverage?

Influential?	Unpaid bills?
True	True
False	False
null	

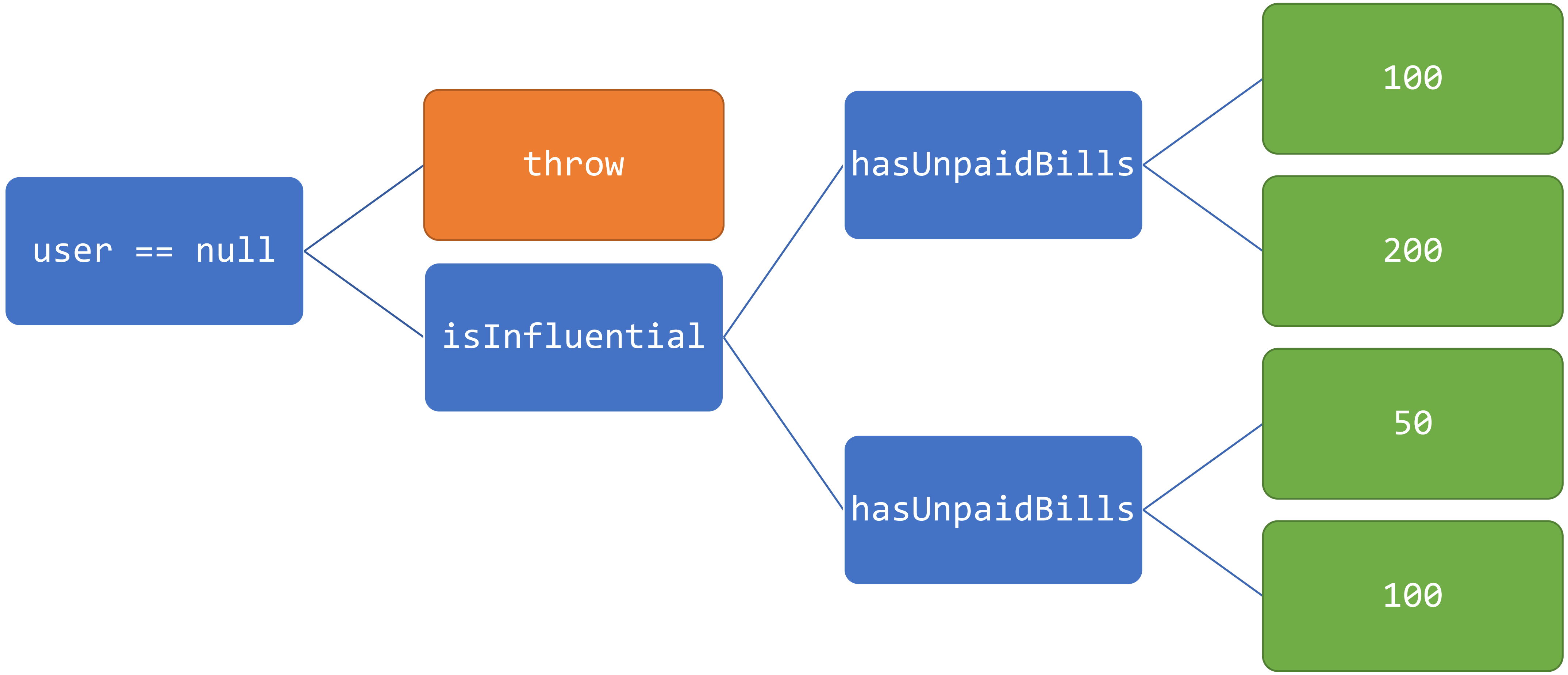
```
int getPriority(User user)
  if (user == null) throw ...;
  int priority = 100;
  if (user.isInfluential()) priority += 100;
  if (user.hasUnpaidBills()) priority /= 2;
  return priority;
}
```

Branch coverage?

☺?

```
int getPriority(User user) {
  if (user == null) throw ...;
  int priority = 100;
  if (user.isInfluential()) priority += 100;
  if (user.hasUnpaidBills()) priority /= 2;
  return priority;
}
```

Program paths



Path coverage

Influential?	Unpaid bills?
True	True
False	False
null	

```
int getPriority(User user) {
  if (user == null) throw ...;
  int priority = 100;
  if (user.isInfluential()) priority += 100;
  if (user.hasUnpaidBills()) priority /= 2;
  return priority;
}
```

Path coverage

Influential?	Unpaid bills?
True	True
False	False
null	

```
int getPriority(User user)
    if (user == null) throw ...;
    int priority = 100;
    if (user.isInfluential()) priority += 100;
    if (user.hasUnpaidBills()) priority /= 2;
    return priority;
}
```

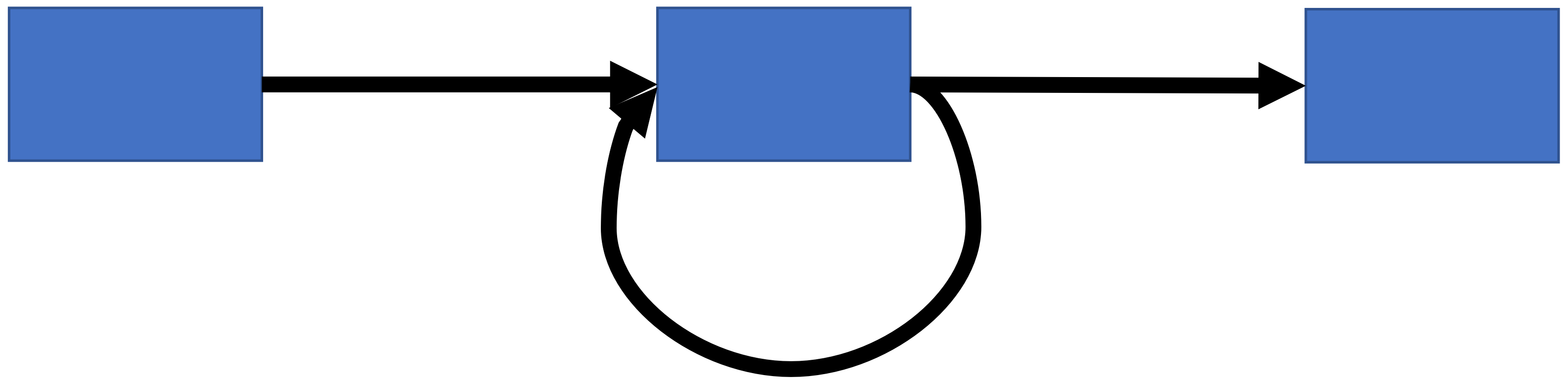
Path coverage

```
int getPriority(User user)
  if (user == null) throw
  int priority = 100;
  if (user.isInfluential()) priority += 100;
  if (user.hasUnpaidBills()) priority /= 2;
  return priority;
}
```

Influential?	Unpaid bills?
True	True
True	False
False	True
False	False
null	

Path coverage

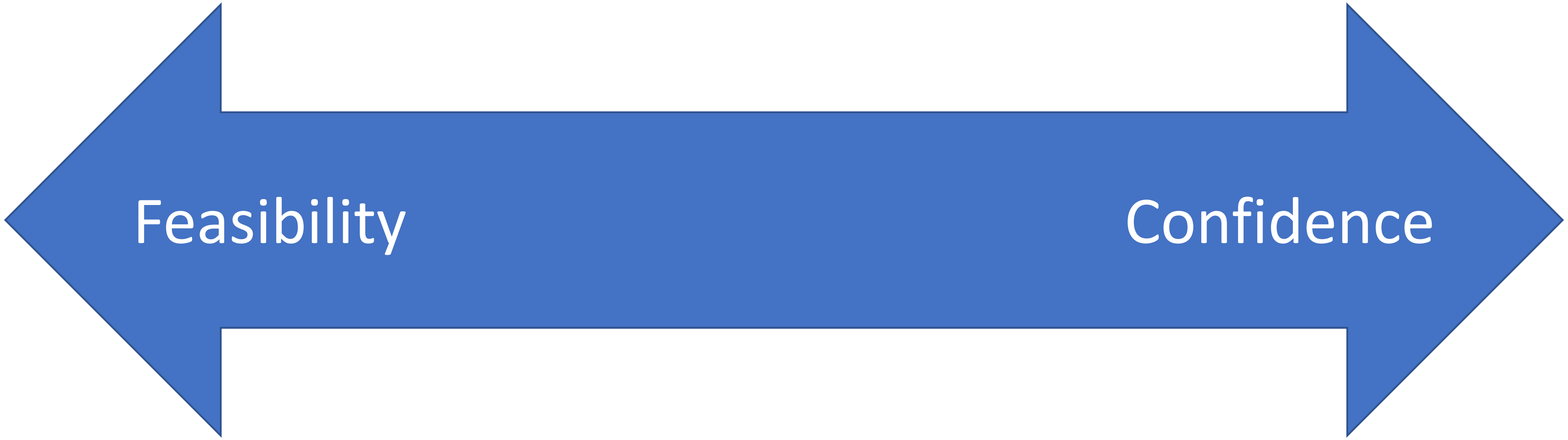
```
while (true) {
  String input = getUserInput();
  if (input.length() < 10) break;
  tellUser("Less than 10 chars please");
}
```



Path coverage

```
if (...) { ... }

if (...) { ... }

if (...) { ... }

if (...) { ... }

if (...) { ... }
```

Coverage trade-offs

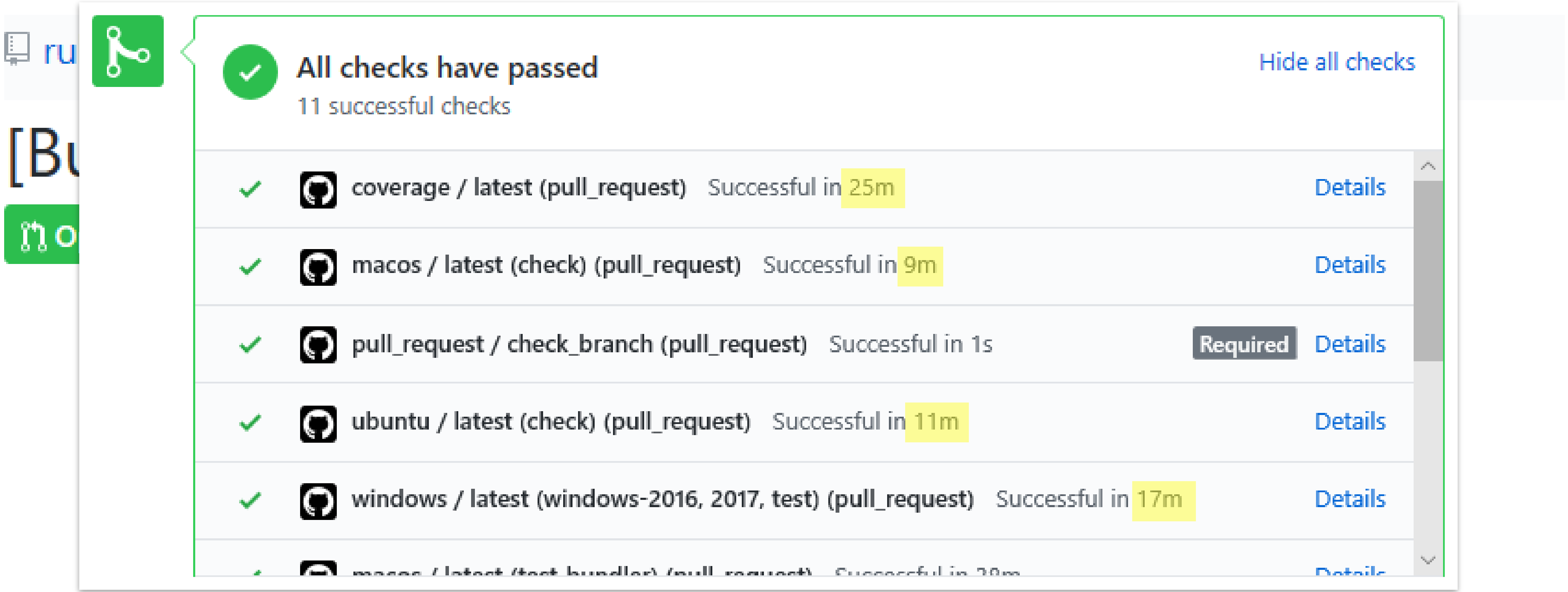


Statement Branch Path

“Coverage”

```
@Test
void coverCode() {
    getPriority(new User(...));
    getPriority(new User(...));
    getPriority(new User(...));
}
```

Performance



ru

[Bu

All checks have passed

11 successful checks

Hide all checks

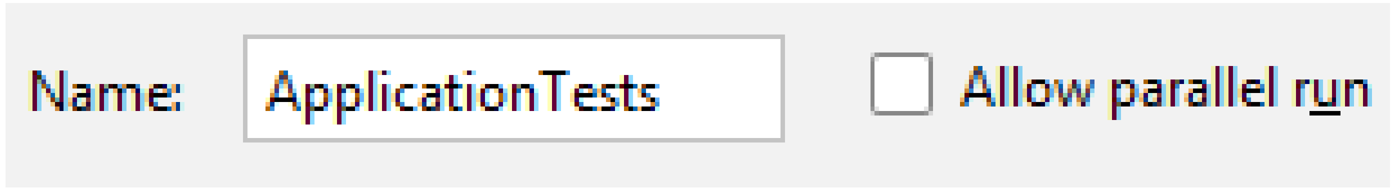
✓	coverage / latest (pull_request)	Successful in 25m	Details
✓	macos / latest (check) (pull_request)	Successful in 9m	Details
✓	pull_request / check_branch (pull_request)	Successful in 1s Required	Details
✓	ubuntu / latest (check) (pull_request)	Successful in 11m	Details
✓	windows / latest (windows-2016, 2017, test) (pull_request)	Successful in 17m	Details

Performance

- How fast does each test run?
- How fast do all tests run?
- How many tests run?

Fast Tests

- Avoid timeouts
  - Use callbacks instead
- Tests must be independent
  - Enables parallelism



Name: ApplicationTests

☐ Allow parallel run

Running a subset of tests

```
@Tag("fast")
@Test
void cannotAddNullUser() { ... }

@Test
void endToEndFriendAdd() { ... }
```