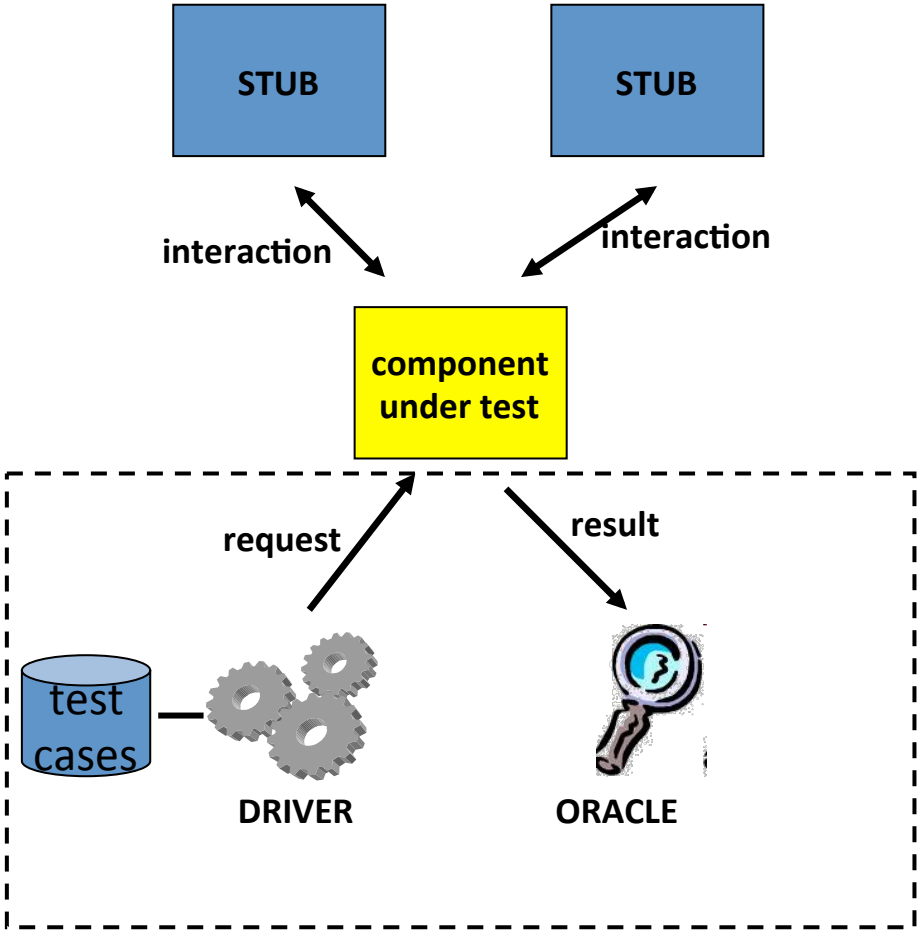
Mocking con JMock

Fabrizio Pastore
www.fabriziopastore.com

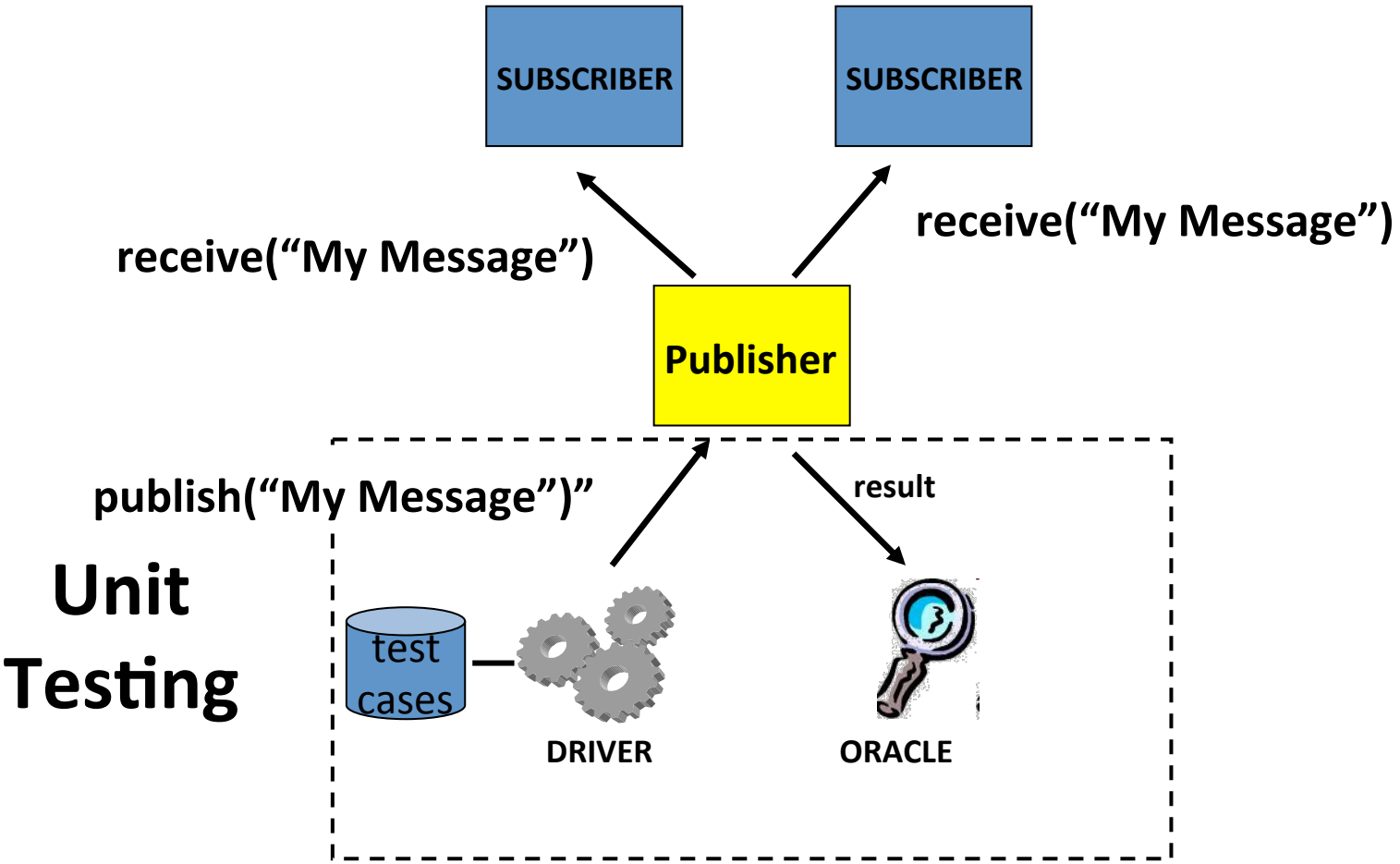
Quando Serve uno Stub?

Quando Serve uno Stub?

Unit Testing



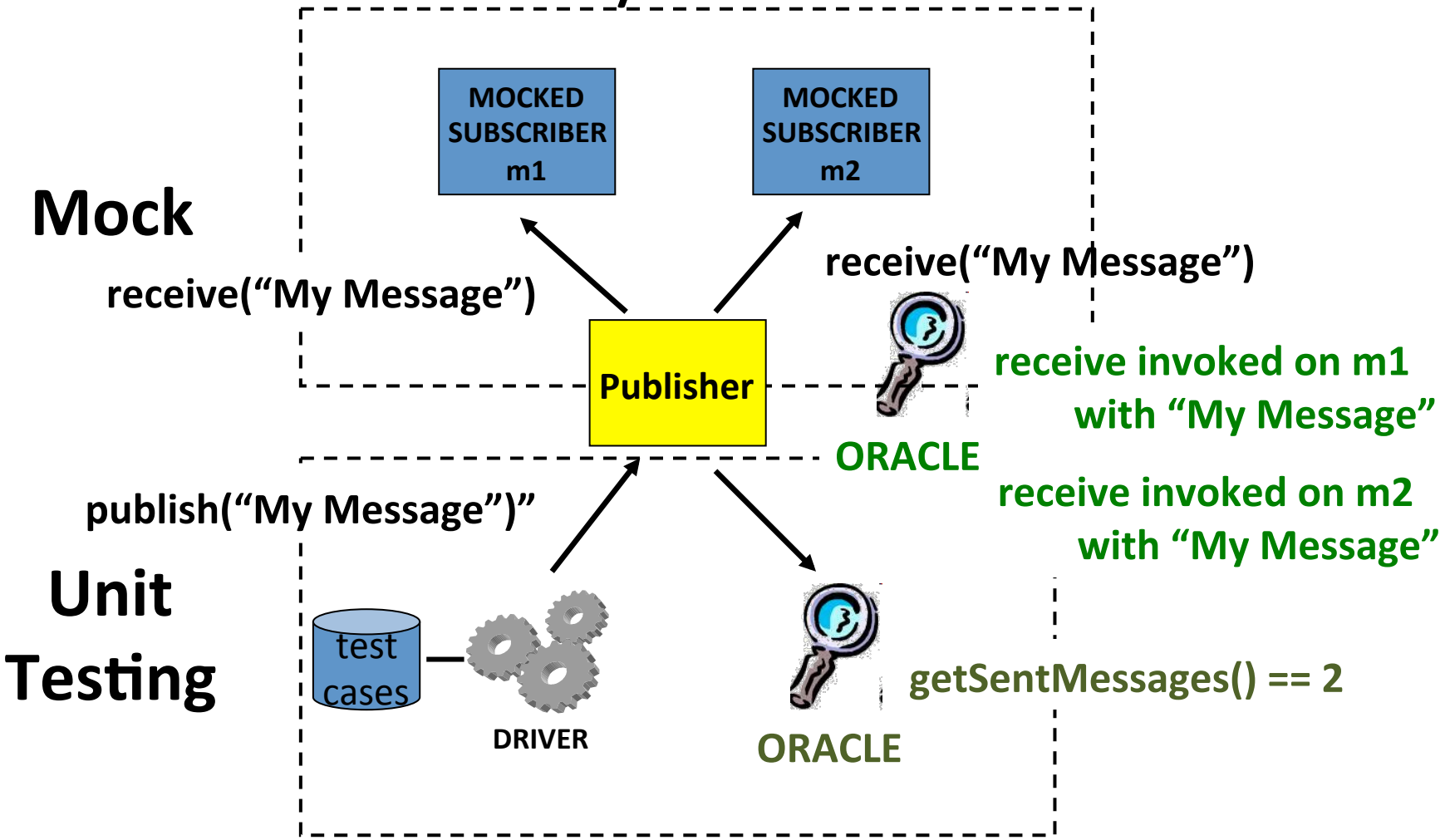
Ad esempio per Testare un sistema Publish / Subscribe



A cosa servono i mock?

- Facilitano l'implementazione degli stub e la verifica della correttezza delle interazioni tra il componente testato e gli stub

Ad esempio per Testare un sistema Publish / Subscribe



Expectations

invocation-count (mock-object).method(argument-constraints);
inSequence(sequence-name);
when(state-machine.is(state-name));
will(action);
then(state-machine.is(new-state-name));

invocation-count

oneOf	The invocation is expected once and once only.
exactly(*n*).of	The invocation is expected exactly *n* times. Note: oneOf is a convenient shorthand for exactly(1).
atLeast(*n*).of	The invocation is expected at least *n*.
atMost(*n*).of	The invocation is expected at most *n* times.
between(*min*, *max*).of	The invocation is expected at least *min* times and at most *max* times.
allowing	The invocation is allowed any number of times but does not have to happen.
ignoring	The same as allowing. Allowing or ignoring should be chosen to make the test code clearly express intent.
never	The invocation is not expected at all. This is used to make tests more explicit and so easier to understand.

Match Objects or Methods

Matcher	Description
aMethod(m)	A method matching the Matcher m is expected.
method(r)	A method with a name that matches the regular expression r is expected.
with(m_1, m_2, ... m_n)	The parameters must match matchers m_1 to m_n.
withNoArguments()	There must be no parameters

- To allow invocations of any bean property getter on any mock object:
  - allowing (any(Object.class)).method("get.*").withNoArguments();
- Allow a method to be called once on any one of a collection of mock objects:
  - oneOf (anyOf(same(o1),same(o2),same(o3))).method("doSomething");

Default argument Matchers

Name	Description
equal(*n*)	The argument is equal to *n*.
same(*o*)	The argument is the same object as *o*.
any(Class<T> type)	The argument is any value. The type argument is required to force Java to type-check the argument at compile time.
a(Class<T> type) an(Class<T> type)	The argument is an instance of *type* or a subclass of *type*.
aNull(Class<T> type)	The argument is null. The type argument is required to force Java to type-check the argument at compile time.
aNonNull(Class<T> type)	The argument is not null. The type argument is required to force Java to type-check the argument at compile time.
not(*m*)	The argument does not match the Matcher *m*.

+ import static org.hamcrest.Matchers.*

Actions

will(returnValue(v))	Return v to the caller.
will(returnIterator(c))	Return a new iterator over collection c on each invocation.
will(returnIterator(v_1, v_2, ..., v_n))	Return a new iterator over elements v_1 to v_n on each invocation.
will(throwException(e))	Throw e to the caller.
will(doAll(a_1, a_2, ..., a_n))	Do all actions a_1 to a_n on every invocation.

Come sono gestite le chiamate

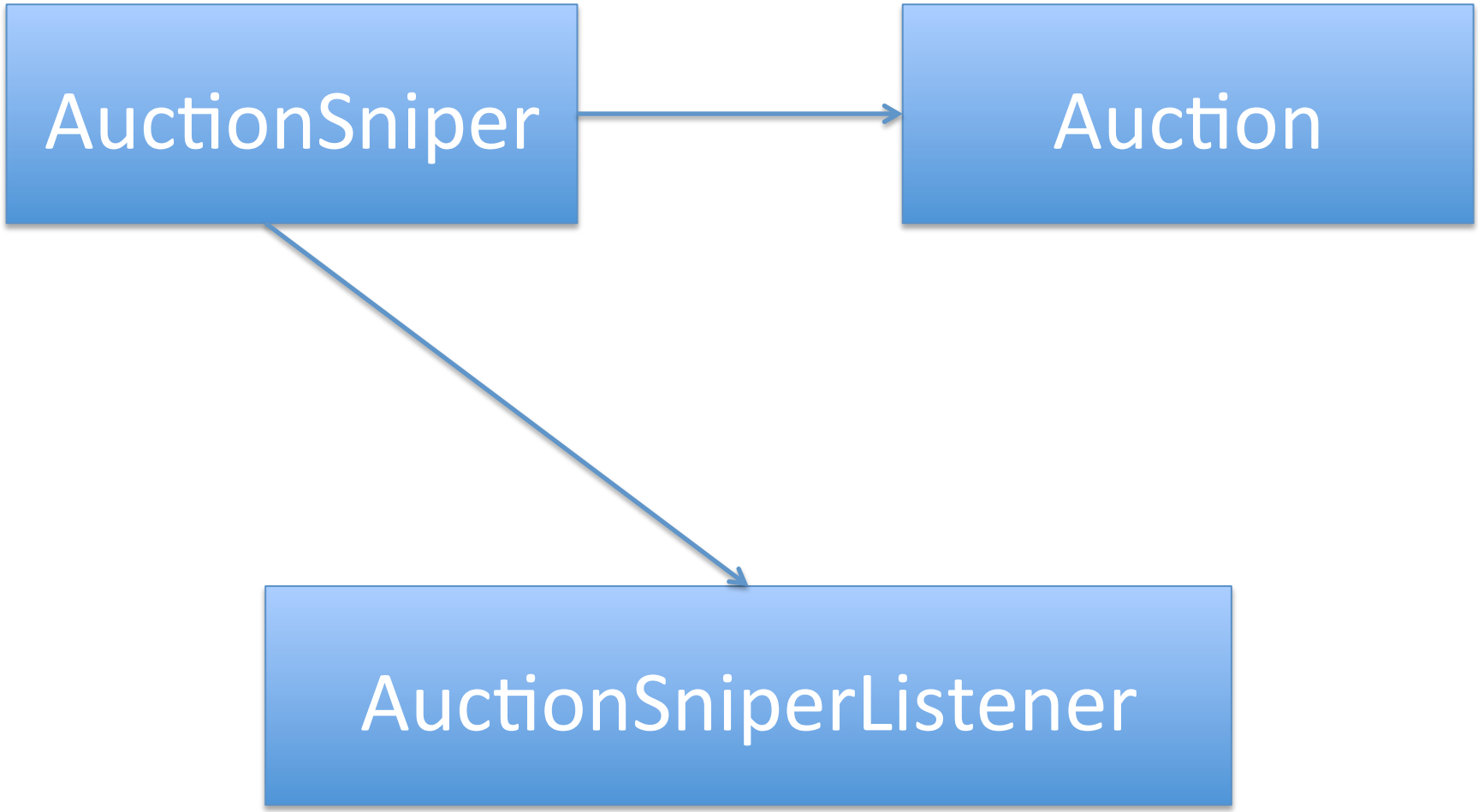
- Quando un metodo è invocato su un mock object
- JMock cerca tra le expectations dalla più vecchia alla più nuova
  - quella trovata potrebbe non essere più usata in futuro (e.g. oneof)
  - ATTENZIONE a: *allowing* / *ignoring* / *atLeast(n)*
    - potrebbero mascherare le chiamate successive
  - non creare più expectations su uno stesso metodo con gli stessi matchers, usare *onConsecutiveCalls*
  - *se è importante l'ordine delle chiamate usare sequence o state-machine*

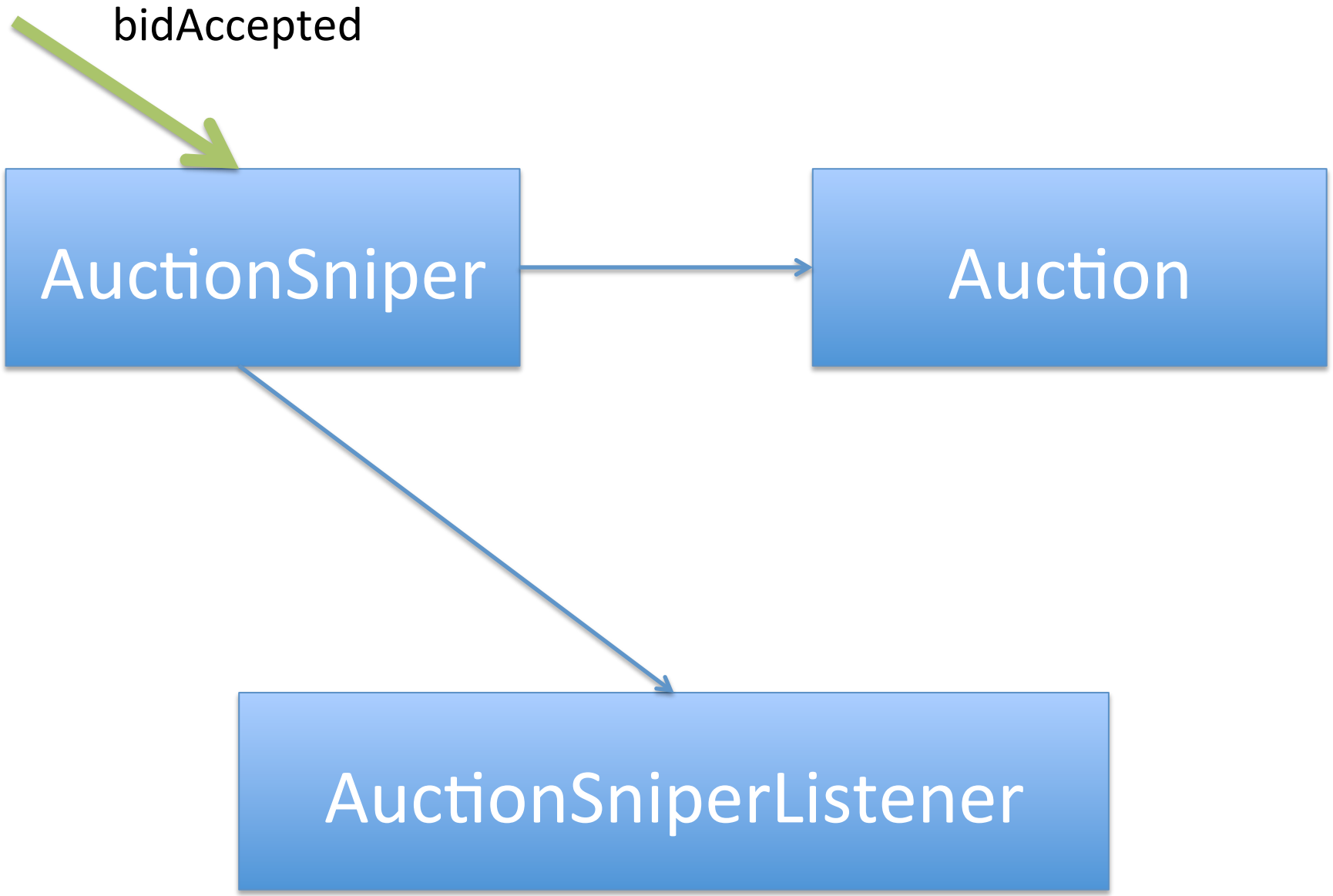
Creare Casi Eccezionali

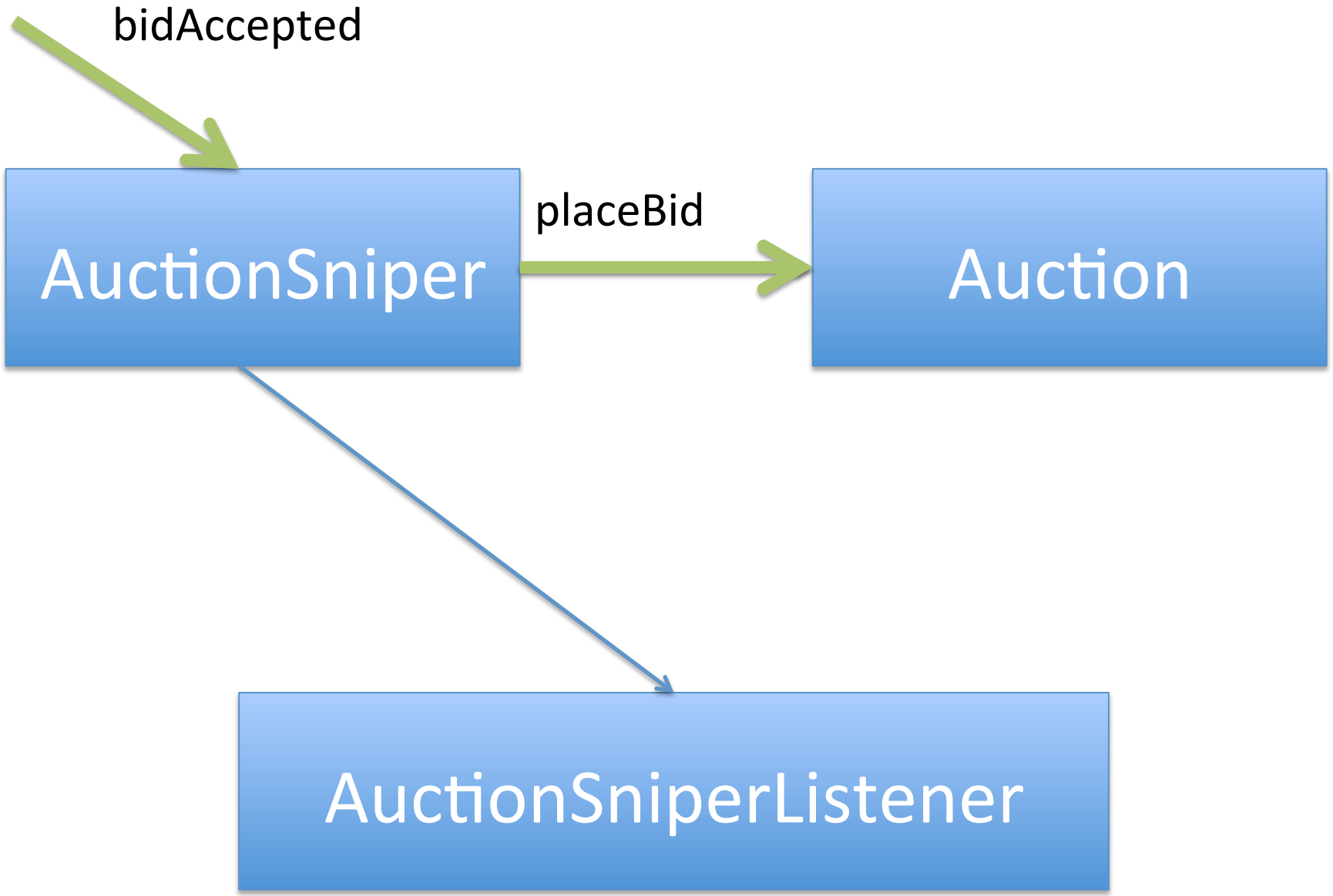
- will(throwException(exception))
- Esempio:
  - AuctionSniperTest
  - webapp/src/test/java petstore.PetStoreTest

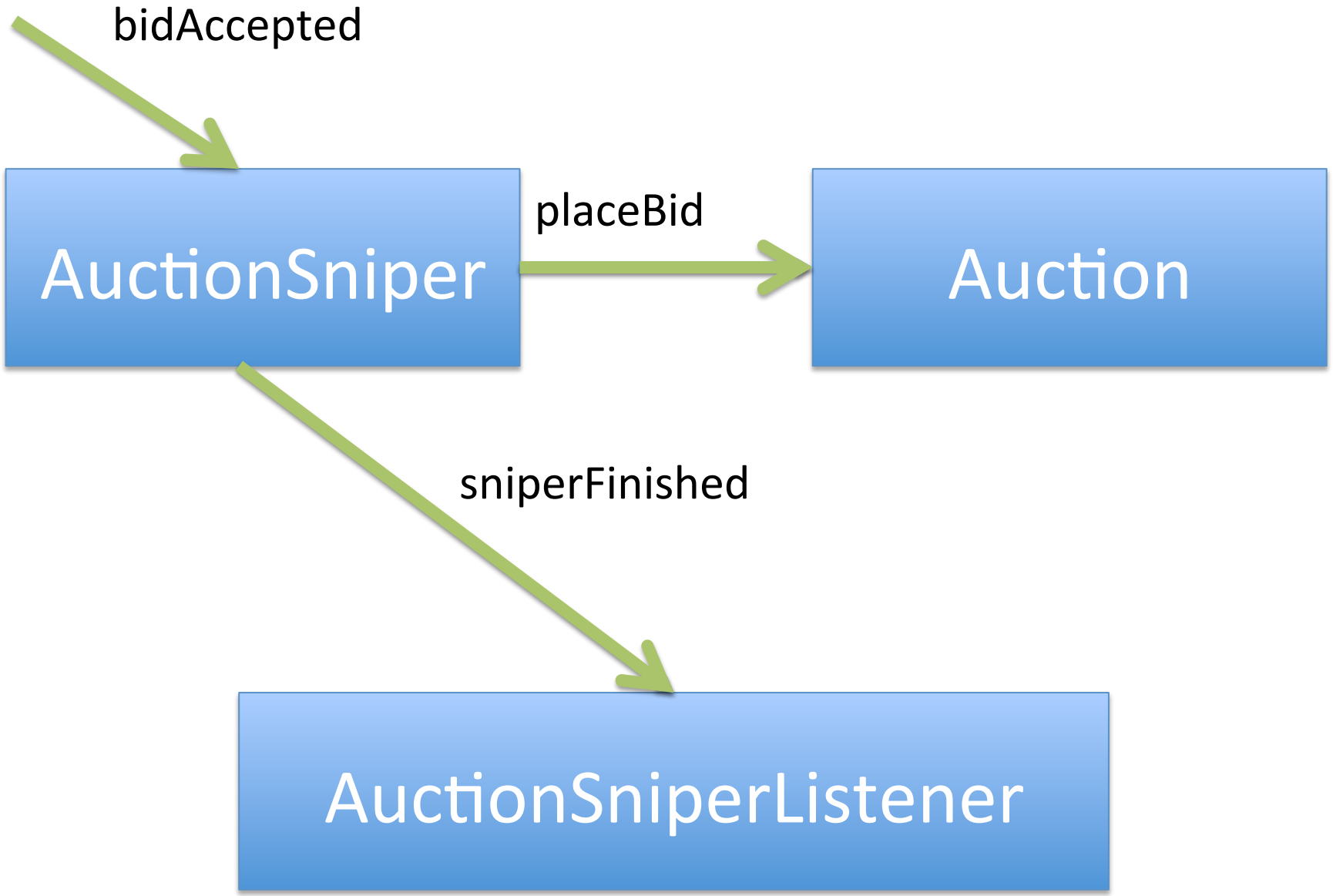

AuctionSniper

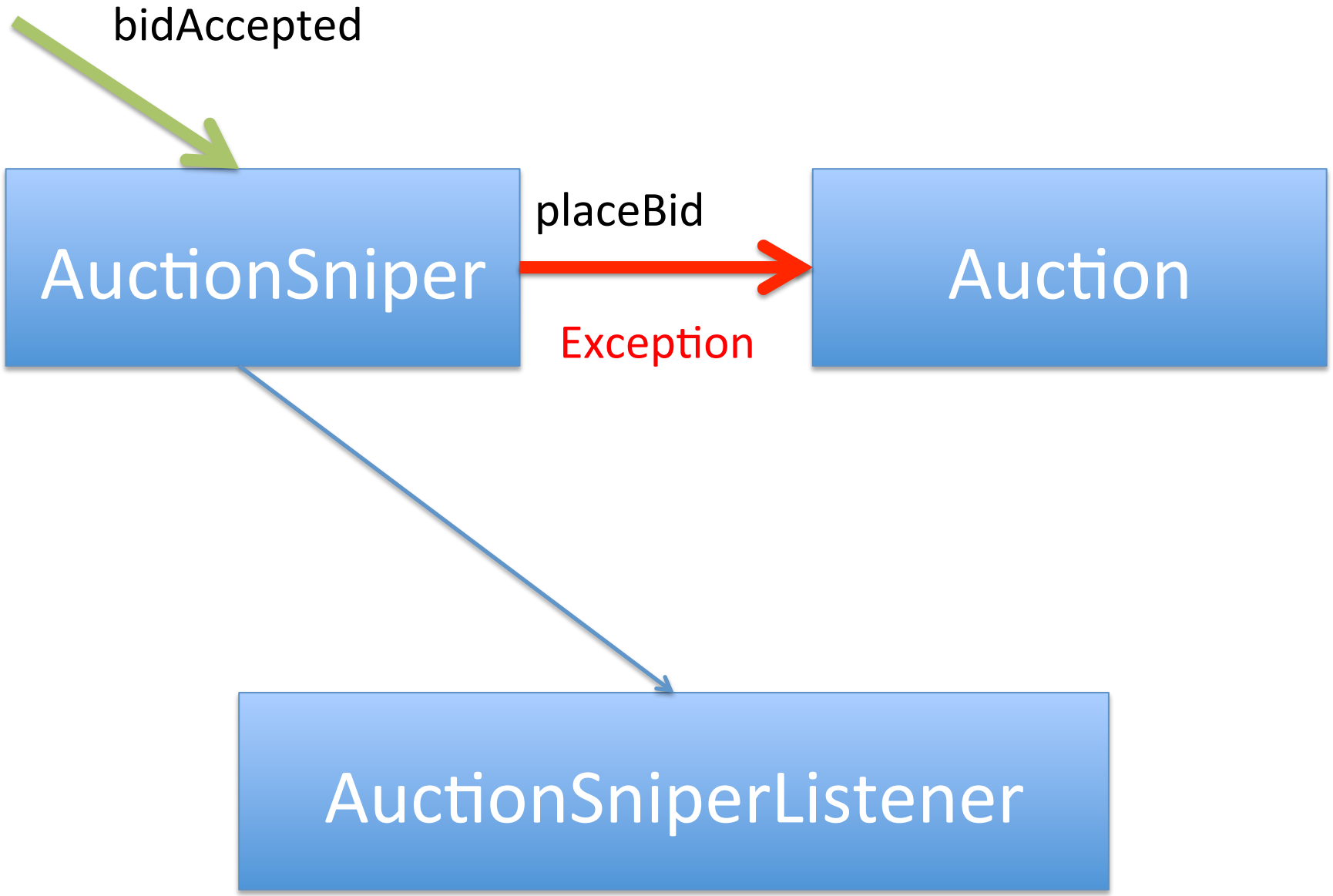

AuctionSniper
Auction

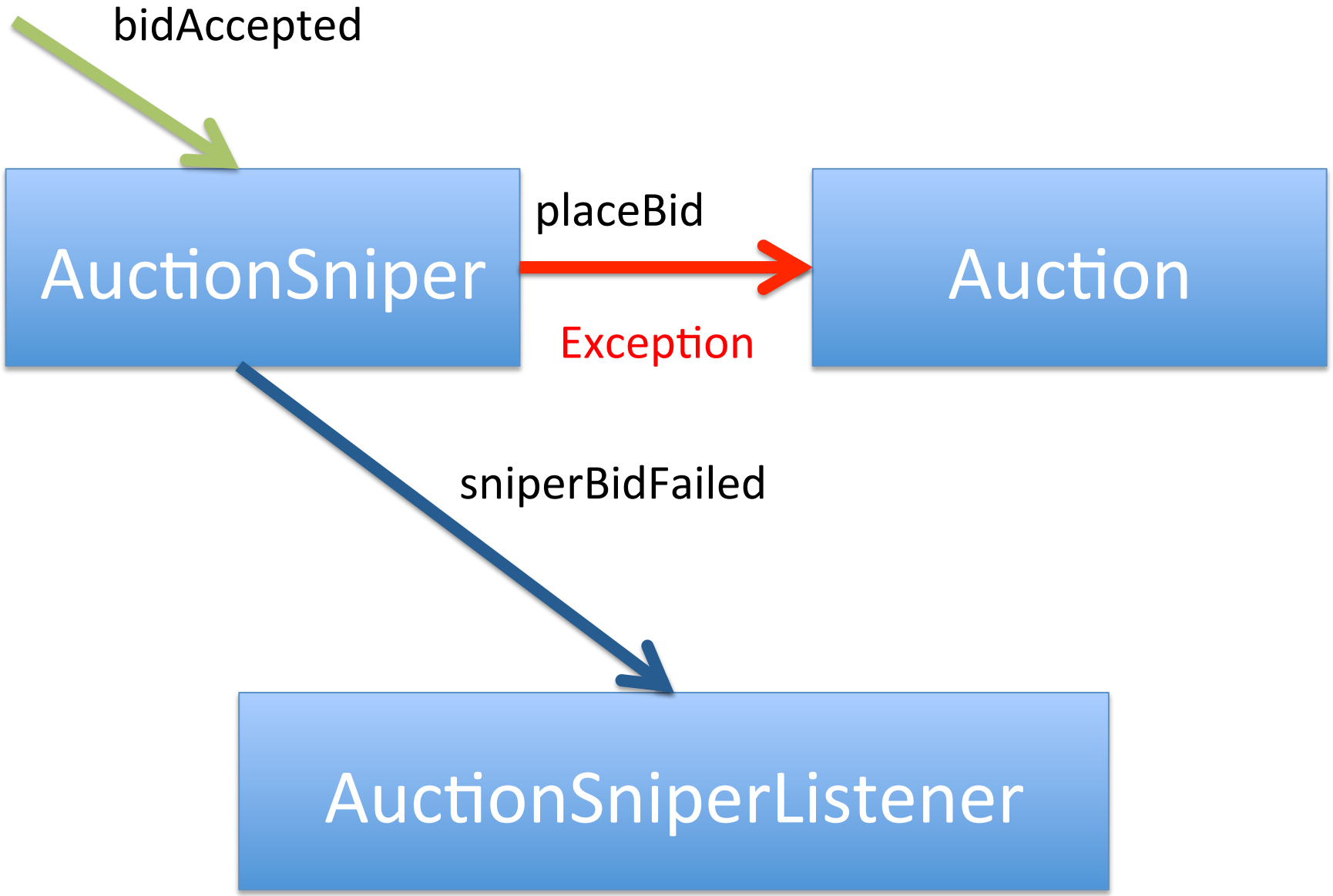

AuctionSniper
Auction
AuctionSniperListener


bidAccepted
AuctionSniper
Auction
AuctionSniperListener


bidAccepted
AuctionSniper
placeBid
Auction
AuctionSniperListener


bidAccepted
AuctionSniper
placeBid
Auction
sniperFinished
AuctionSniperListener


bidAccepted
AuctionSniper
placeBid
Exception
Auction
AuctionSniperListener


bidAccepted
AuctionSniper
placeBid
Exception
Auction
sniperBidFailed
AuctionSniperListener

ignoring(mock);

Return Type	***Default Value***
boolean	false
numeric type	zero
String	"" (empty string)
Array	Empty array
Mockable type	A mock that is ignored
Any other type	null

State Machine

- Encode the stub as a state machine
- Declare the state machine
  - final States *fsa-name* = context.states(”*fsa-name*").startsAs("*initial-state*");
- Use the state machine

when(state-machine.is("state-name")); Constrains the last expectation to occur only when the state machine is in the named state.

when(state-machine.isNot("state-name")); Constrains the last expectation to occur only when the state machine is not in the named state.

then(state-machine.is("state-name")); Changes the state of *state-machine* to the named state when the invocation occurs.

State Machine - Example

Example

```
final States file = context.states(”file").startsAs(”closed");
oneOf (myFile).open(); then(file.is(”opened"));
oneOf (myFile).readChar(); when(file.is(”opened")); will(returnValue(‘c’));
oneOf (myFile).readLine(); when(file.is(”opened")); will(returnValue(“line”));
oneOf (myFile).readChar(); when(file.is(”opened")); will(returnValue(‘c’));
oneOf (myFile).close(); then(file.is(”closed"));
```

```
invocation-count (mock-object).method(argument-constraints);
    inSequence(sequence-name);
    when(state-machine.is(state-name));
    will(action);
    then(state-machine.is(new-state-name));
```