Classes and Objects

COMP1400/INFS1609

Week 8

Object-oriented Programming

Abstraction

- Dividing the program into chunks at different levels of detail.

Encapsulation

- Each chunk hides its implementation details from outsiders.

Public/private

Each object has:

- a public interface that describes how it can be used
- a private implementation that describes how it works

Classes

Objects have types called Classes.

Classes represent all objects of a certain kind.

All objects in a class share a common structure but have different details.

Example

"Car" is a class.

All cars have data:
- colour
- engine capacity, etc.

and methods:
- drive
- park, etc.

Example

My car is an object.
It is an instance of the class Car.

My car has data with specific values:
- colour = yellow
- engine capacity = 1.5 litre

and methods:
- drive
- park, etc.

Static data and methods

You usually need a specific instance of a class (an object) to access data and methods.

Eg: you can't drive the class "Car" or ask what colour "Car" is.

However some methods and data belong to all cars and can be run on the class. These are called "static".

Java Class Library

Java comes with a large collection of classes for common object types.

You can browse the library online at:

http://docs.oracle.com/javase/6/docs/api/index.html?overview-summary.html

Random

The Random class:

http://docs.oracle.com/javase/6/docs/api/java/util/Random.html

Implements a random number generator.

JCL docs

Things to notice in the docs for Random:

- The package - java.util.Random
- The constructors
- The methods

Creating objects

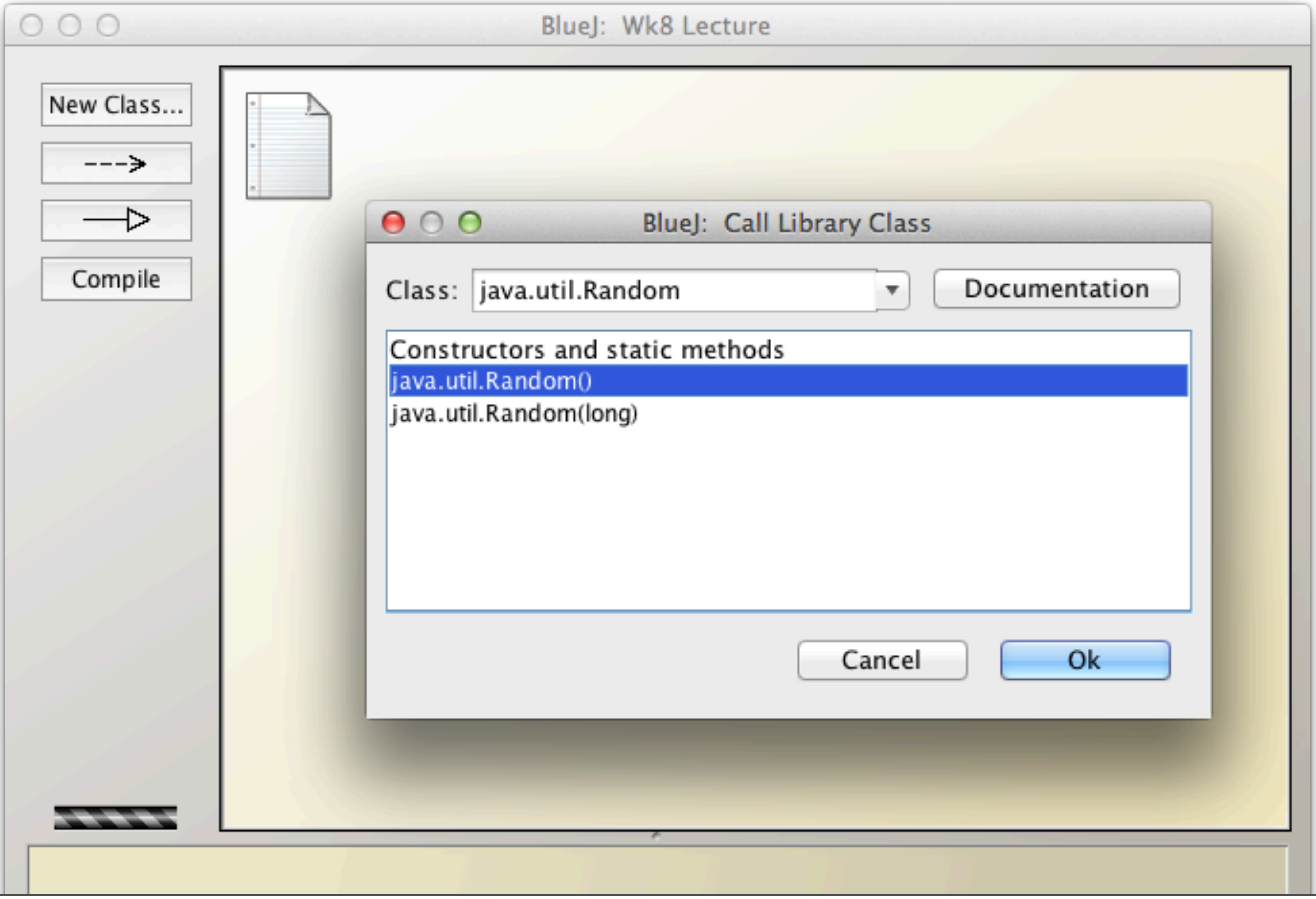
To use an object we must first create an instance of the class.

We do this by calling one of the constructors.

The constructor creates the object and initialises it so it is ready to use.

JCL in BlueJ

Select Tools > Use Library Class...



Class: java.util.Random

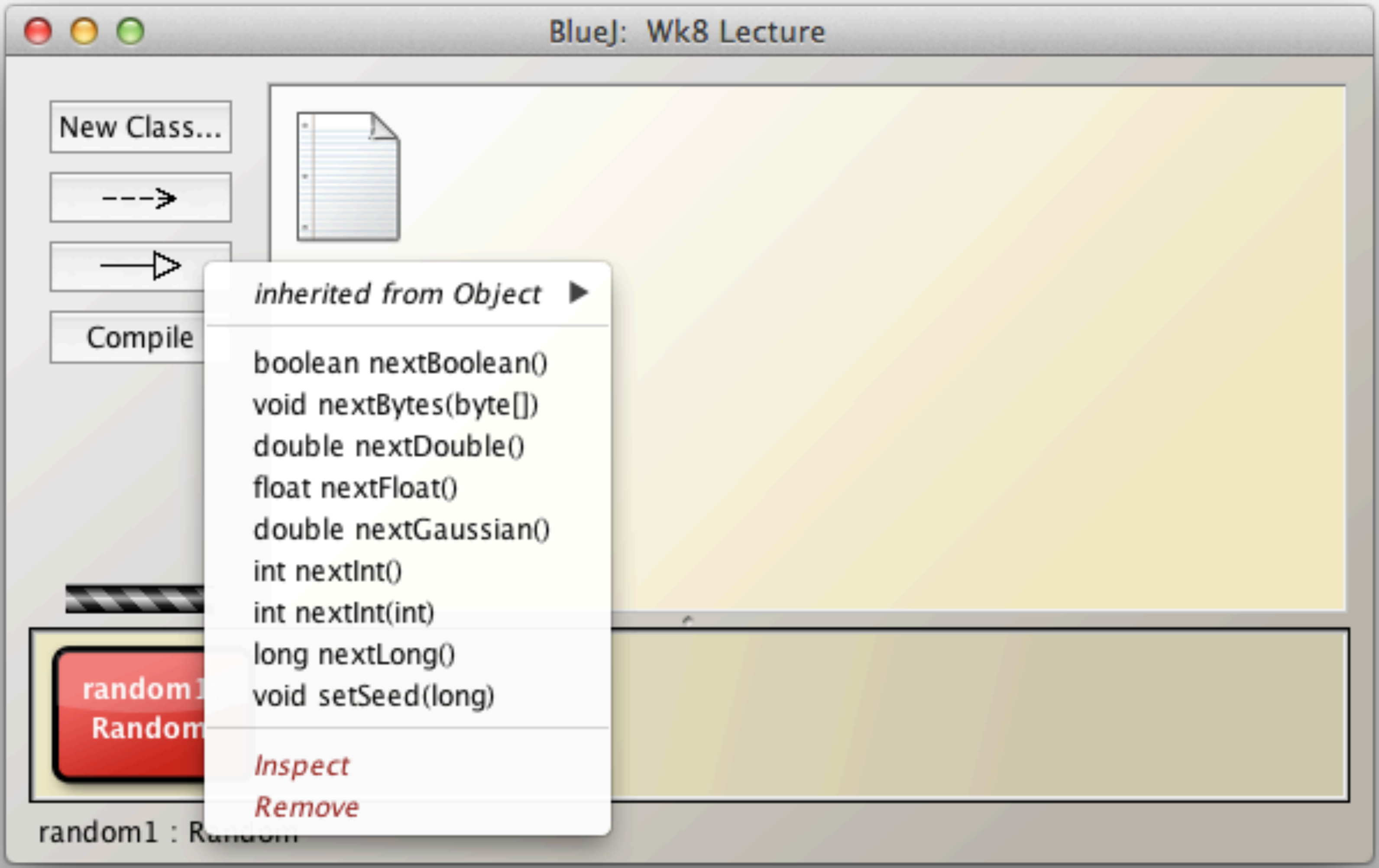
Constructors and static methods

java.util.Random()

java.util.Random(long)

Methods

Each Random object has a number of methods:



```
boolean nextBoolean()
void nextBytes(byte[])
double nextDouble()
float nextFloat()
double nextGaussian()
int nextInt()
int nextInt(int)
long nextLong()
void setSeed(long)
```

Importing

If we want to use a JCL class in our code, we need to import it first.

Syntax:

```
import java.util.Random;
```

‘import’ keyword

Full class name.

Calling constructors

To create an object in code we call the constructor as:

```
new Random();
```

or:

```
new Random(100);
```

‘new’ keyword

class

parameters

Classes are types

Classes are types and can be used in the same way as primitive types like `int` and `double` to create variables.

```
int years = 100;

Random rng = new Random(100);
```

State

Objects have state -- internal private data which describe their current configuration.

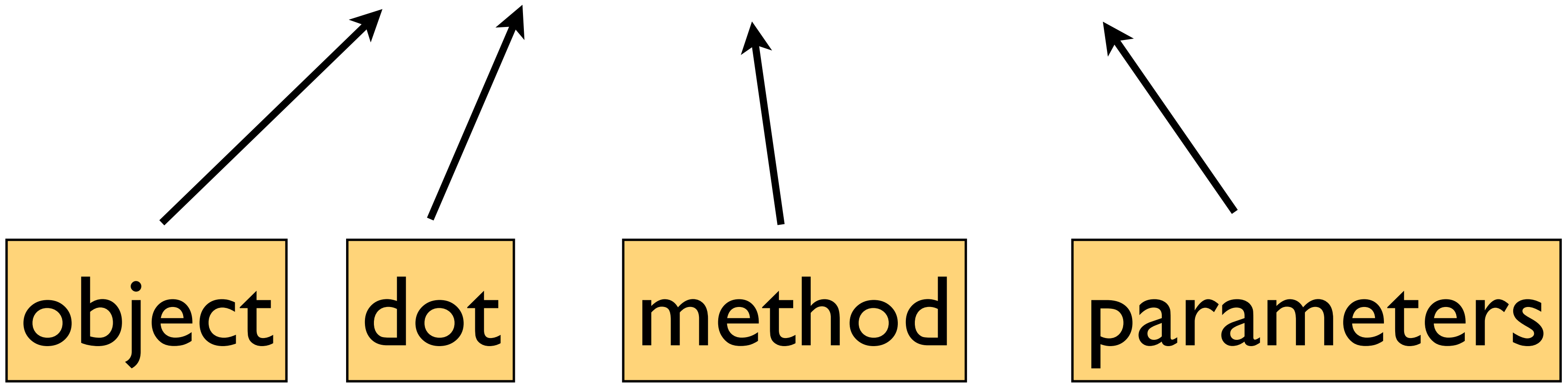
E.g. the state of a Car object would include:

- the amount of fuel it has left,
- the number of miles it has been driven,
- what gear it is in,
- etc....

Calling methods

To call methods on an object:

```
Random rng = new Random(100);
int roll = rng.nextInt(6);
```



Accessing state

An object's state is private (encapsulated).

The class may provide public methods to access state and manipulate it in proscribed ways.

Eg:

```
    rng.setSeed(100);
```

Random seed

The seed of a random number generator determines the values it creates. Two Random objects with the same seed produce the same sequence:

```
Random rng1 = new Random(100);
Random rng2 = new Random(100);
```

References

When we create an object it is allocated as a block of data in memory. The value the constructor returns is a reference to that block.

A reference is like an address. It is a piece of information that tells us where the object is.

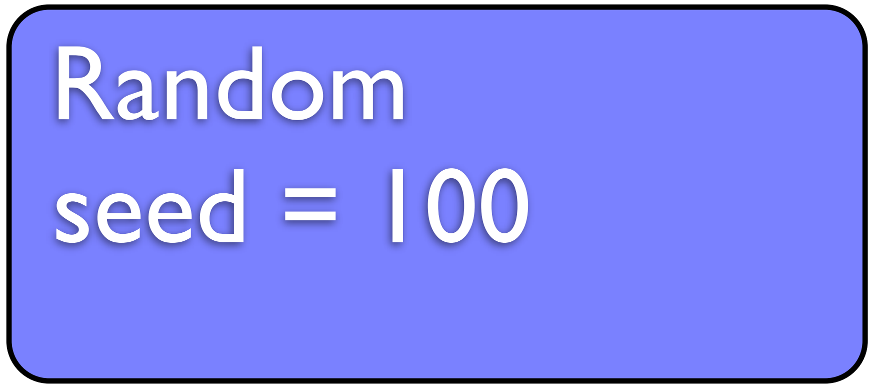
Reference

When we assign an object variable to another variable, we copy the reference **not** the object.

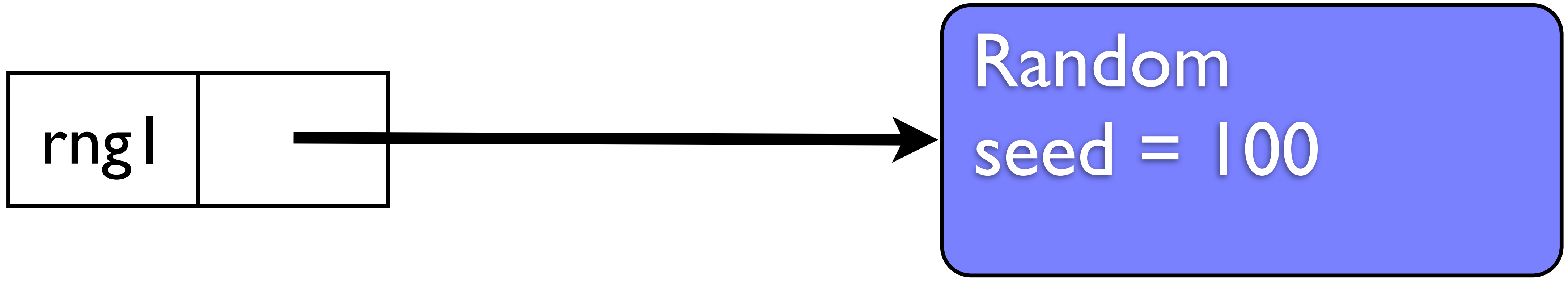
```
Random rng1 = new Random(100);

Random rng2 = rng1;

// both now refer to the same
// object
```

```
Random rng1 = new Random(100);

Random rng2 = new Random(110);

rng2 = rng1;

rng2.setSeed(50);
```

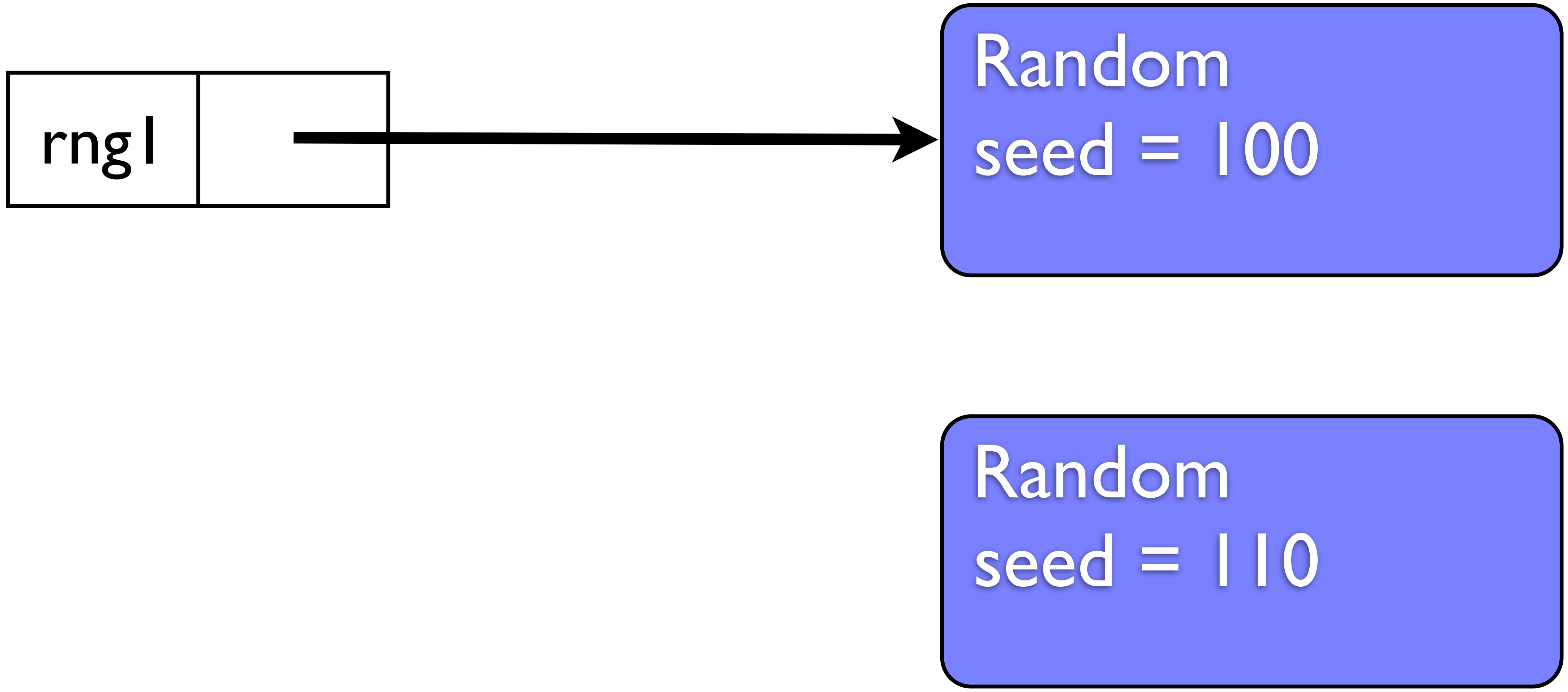
```
Random rng1 = new Random(100);

Random rng2 = new Random(110);

rng2 = rng1;

rng2.setSeed(50);
```



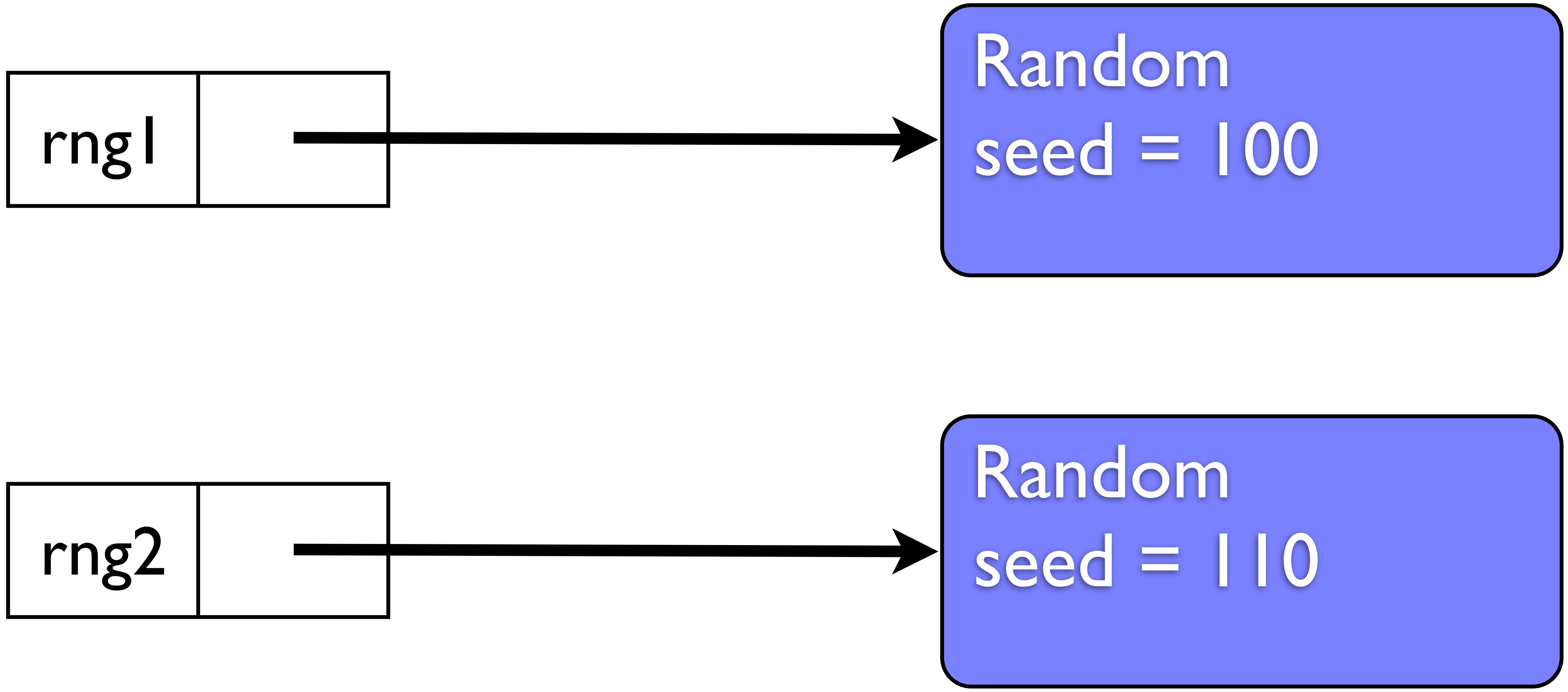
```
Random rng1 = new Random(100);
Random rng2 = new Random(110);
rng2 = rng1;
rng2.setSeed(50);
```



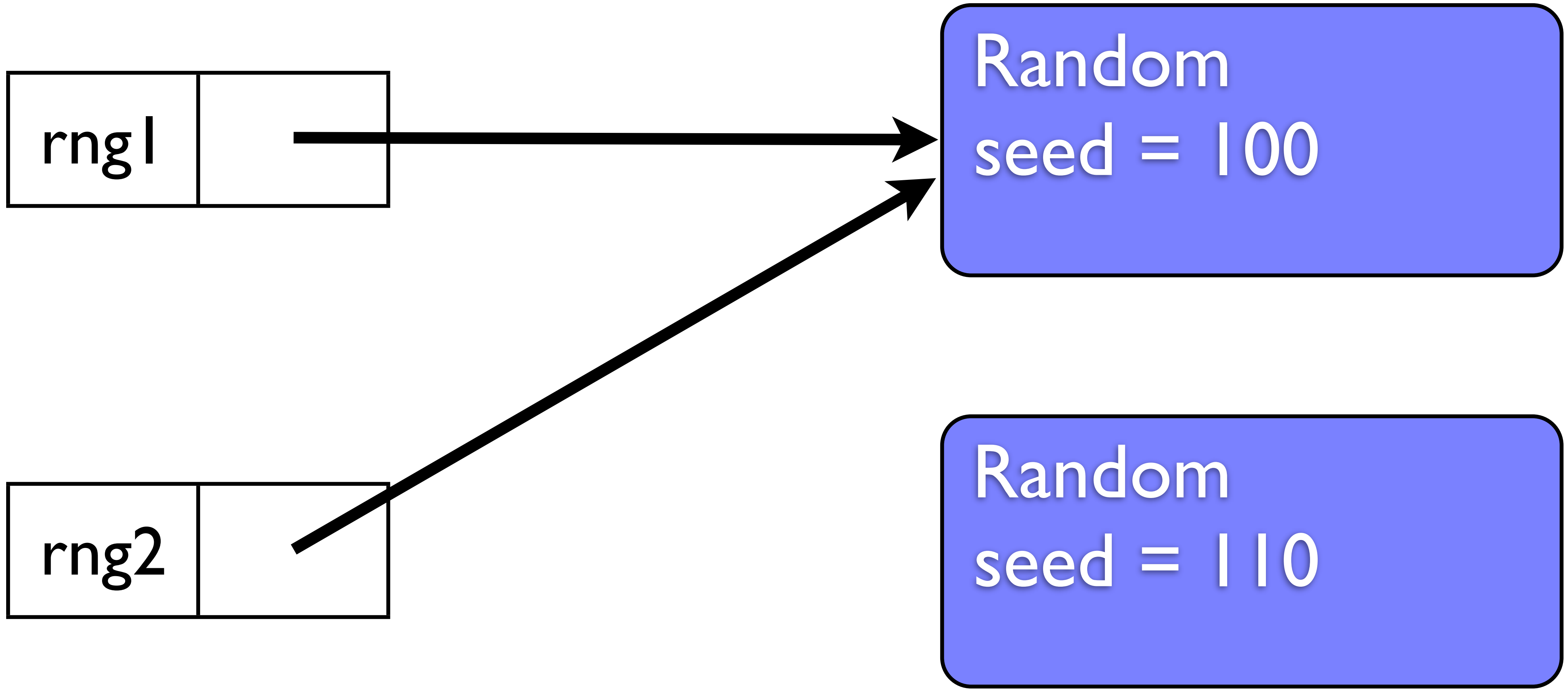
```
Random rng1 = new Random(100);
Random rng2 = new Random(110);
rng2 = rng1;
rng2.setSeed(50);
```



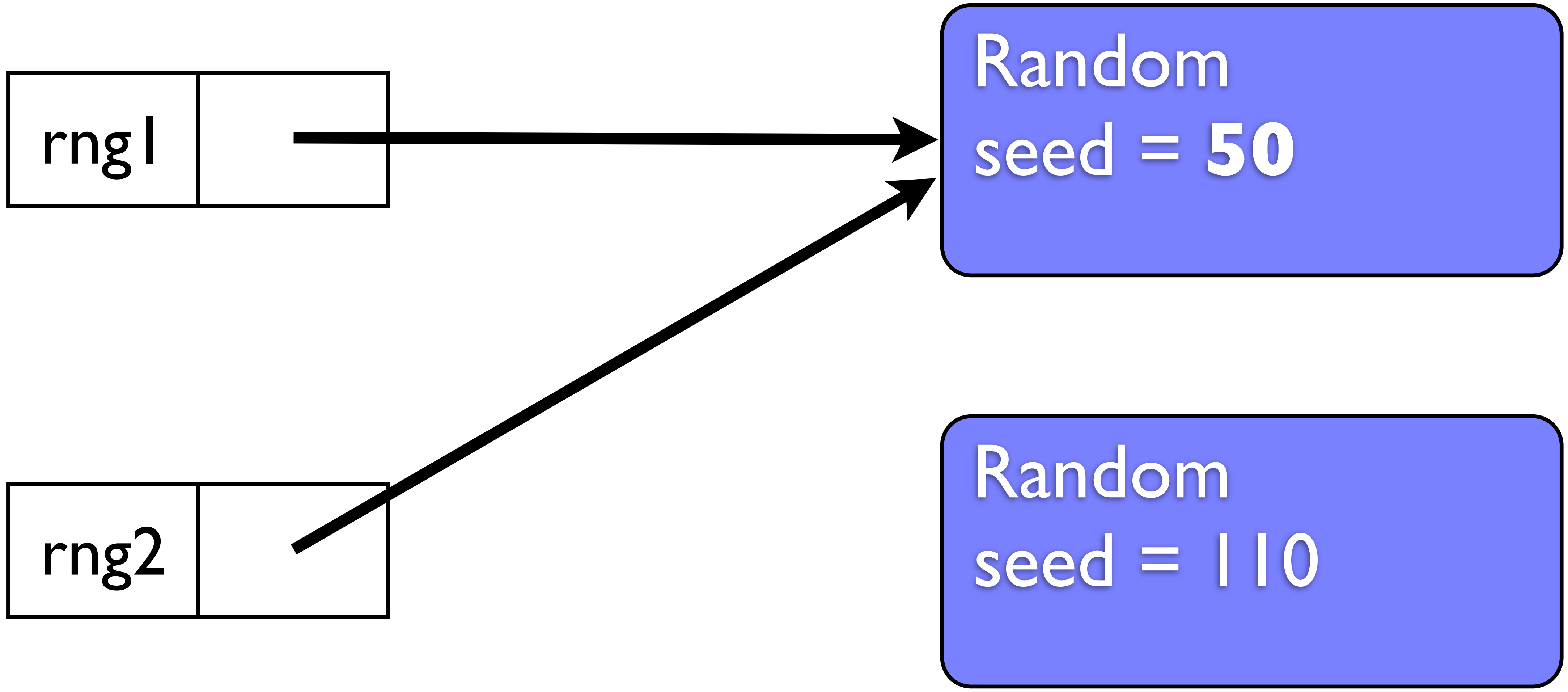
```
Random rng1 = new Random(100);
Random rng2 = new Random(110);
rng2 = rng1;
rng2.setSeed(50);
```



```
Random rng1 = new Random(100);
Random rng2 = new Random(110);
rng2 = rng1;
rng2.setSeed(50);
```

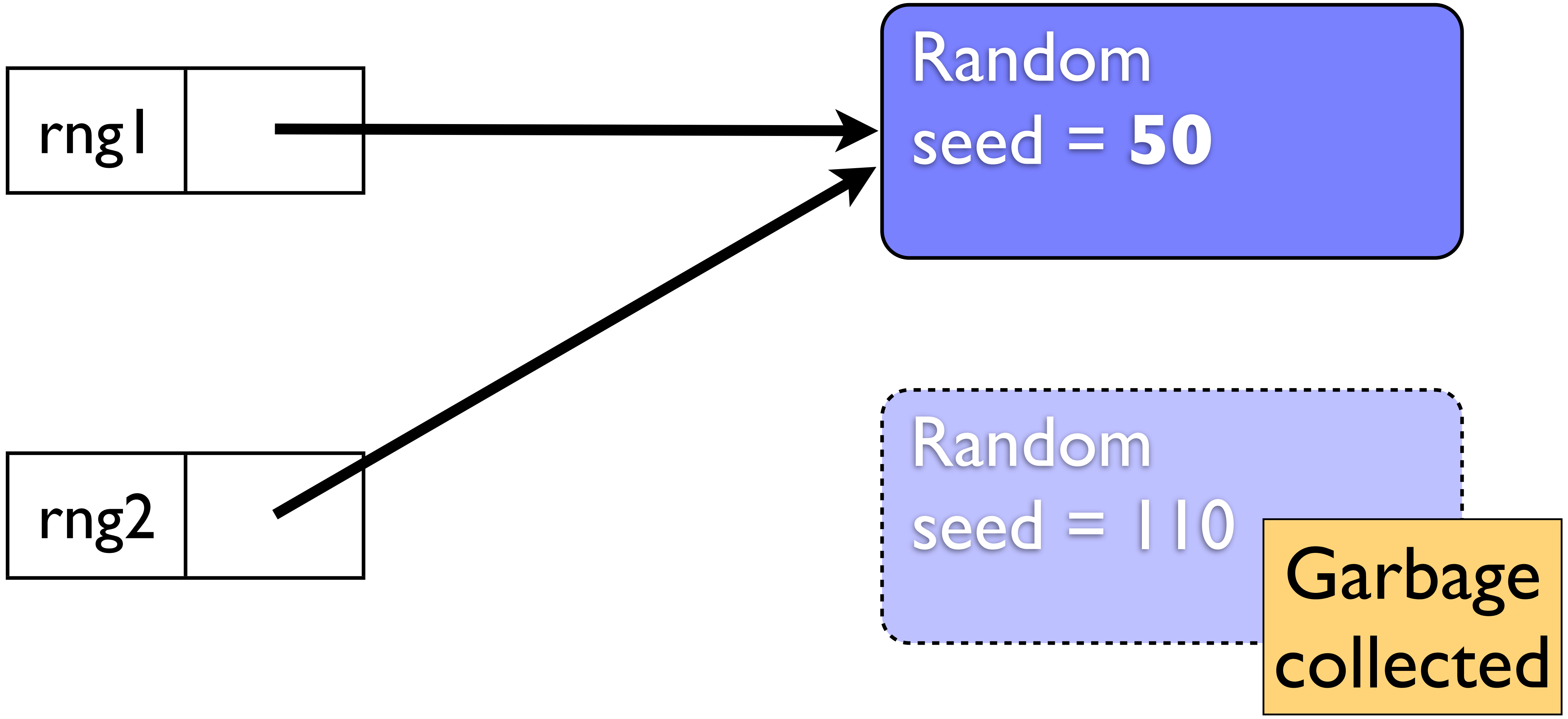


```
Random rng1 = new Random(100);
Random rng2 = new Random(110);
rng2 = rng1;
rng2.setSeed(50);
```

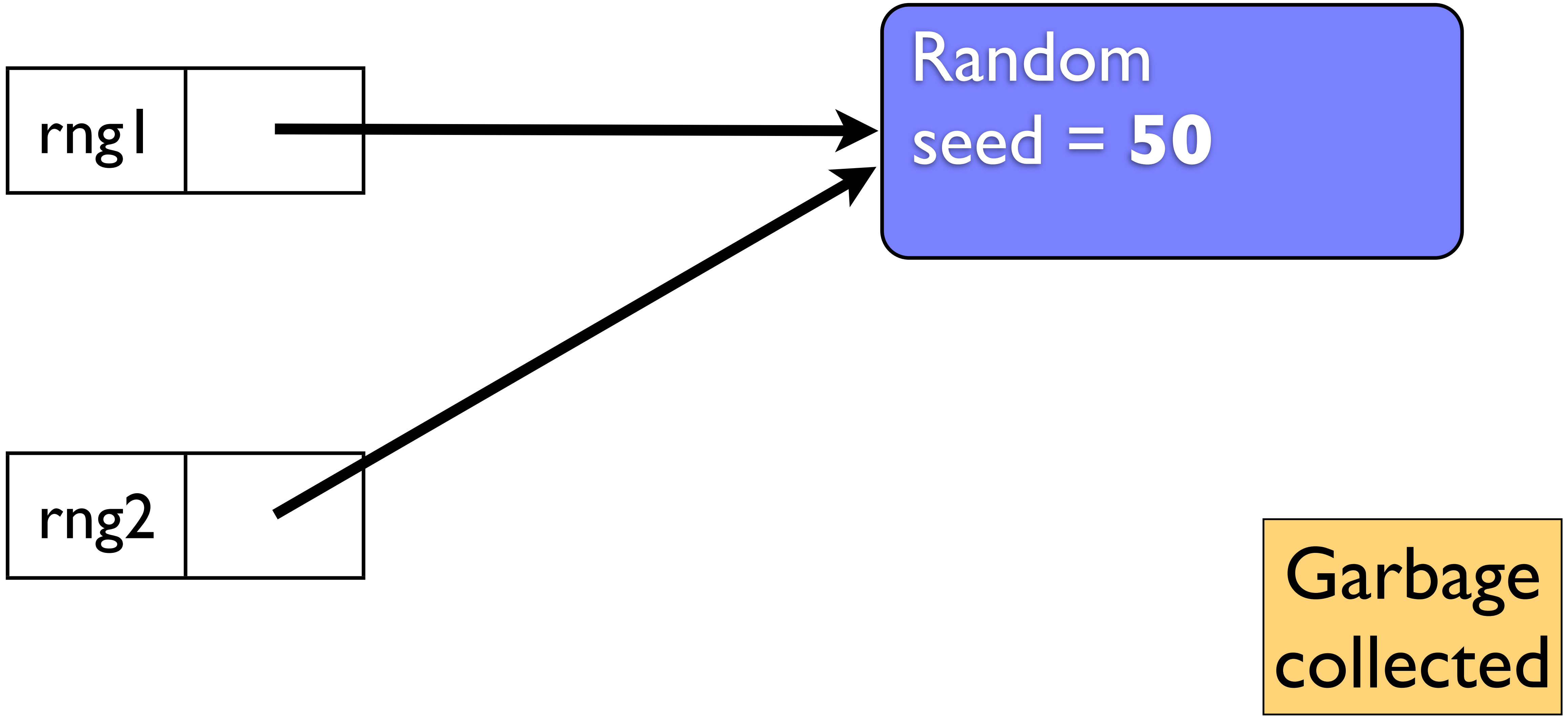


```
Random rng1 = new Random(100);

Random rng2 = new Random(110);

rng2 = rng1;
```

`rng2.setSeed(50);`



```
Random rng1 = new Random(100);
Random rng2 = new Random(110);
rng2 = rng1;
```
`rng2.setSeed(50);`


rng1
rng2
Random
seed = 50
Garbage collected