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/7/docs/api/
index.html?overview-summary.html

Random Numbers

The Random class:

http://docs.oracle.com/javase/7/docs/api/java/util/
Random.html

Implements a random number generator.

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.

Methods

Each Random object has a number of methods:

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);
```

object | dot | method | parameters
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:

```java
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:

```java
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.

```java
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);