Helper Objects Again

A Helper Object Provides Services

- A helper object provides services that other bits of code can use.
- Like a worker with
  - A special area of responsibility
  - A series of tasks they can do.
- Each different task is a method.
  - Each object method needs inputs and produces outputs, just like a helper method.

MyDate Services

- `MyDate` will do the following things:
  - Tell you the day (`getDay`).
  - Tell you the month (`getMonth`).
  - Tell you year (`getYear`).
  - Print on a given `Console` (`print`).
- Not all these services will be required in any one program.
  - But they are there when needed.

What Info Does MyDate Need

- It should know the day, month and year.
  - Stored as instance variables.
  - Given initial values by the constructor.
- It needs a `Console` for printing.
  - Only needed in one method, so not an instance variable.
  - Provided as a parameter to `print`.

Where is the MyDate Code?

- All `MyDate` code must be in a file called `MyDate.java`
- This file should not contain anything else.
- Create with New – Class in eclipse.

Testing MyDate Class

- Write a `main` method in a separate class / file.
  - Call it `Ex1`, for example.
- Both files are in the same eclipse project.
  - eclipse will make sure that both files work well together.
- The `main` method should create `MyDate` objects and call every method.
  - To make sure they all work.
- `MyDate` will then be ready to use in other programs.

Office Analogy

- When we write the code for `MyDate` we are writing instructions on how to make a `MyDate` worker.
  - A robot worker?
- This worker will be able to answer a limited number of questions.
  - The methods.
- We haven't created any workers yet.

Creating a Worker

- `new MyDate` will create a worker who understands about a specific date.
  - The one given in the constructor.
- He can then tell us the day, the month or the year whenever we ask him.
- He can also print this information.

Many Worker

- If our program needs more than one `MyDate` worker.
  - One to know about `now`.
  - One to know about `dateOfBirth`.
- We have to create several workers.
  - We give then different variable names so that we know who is who.

Object Parameters

An Object as a Parameter

- The following program fragment creates a `Console` object and a `MyDate` object.
- The `Console` object is passed to the print method of the `MyDate` object.
- What happens to the `Console` object?

Code

```
    // in main
Console con = new Console();
MyDate d = new MyDate(1, 2, 3);
d.print(con);

    // in MyDate.print
public void print(Console c)
    {
    c.print(String.format(“%02d”, day));
    . . .
    }
```

Addresses Of Objects

- `new Console()` creates a `Console` object in special Java object memory.
- The memory address of this new object is returned.
- `Console con = . . .` stores the address of this object so that it can be located when needed.

A Copy Of The Address

- `d.print(con)` passes this address as a parameter to the `print` method.
- `public void print(Console c)` stores this address in the variable `c`.
- There are now two copies of the address.
  - In `con` and `c`.
- They are the same address, that of the object created by `new`.

Using The Address

- `c.print(String.format("%02d", day));` uses the address stored in `c` to find the object and call the `print` method.
- When the `MyDate` `print` method is finished, the address `c` vanishes.
  - It is a local variable.
- The copy of the address in `main`, stored in `con`, is still there.
  - We can use this `Console` again.

Garbage Collection

- If all copies of an object's address are discarded.
  - The object can't be used.
  - The program does not know where it lives.
- Java will eventually recognise this.
- It will recycle the memory occupied by this object.
  - So that it can be used by another object.