

Notes from Well House Consultants

These notes are written by Well House Consultants and distributed under their Open Training Notes License. If a copy of this license is not supplied at the end of these notes, please visit

*<http://www.wellho.net/net/whcotnl.html>
for details.*

1.1 Well House Consultants

Well House Consultants provides niche training, primarily but not exclusively in Open Source programming languages. We offer public courses at our training centre and private courses at your offices. We also make some of our training notes available under our "Open Training Notes" license, such as we're doing in this document here.

1.2 Open Training Notes License

With an "Open Training Notes License", for which we make no charge, you're allowed to print, use and distribute these notes provided that you retain the complete and unaltered license agreement with them, including our copyright statement. This means that you can learn from the notes, and have others learn from them too.

You are NOT allowed to charge (directly or indirectly) for the copying or distribution of these notes, nor are you allowed to charge for presentations making any use of them.

1.3 Courses presented by the author

If you would like us to attend a course (Java, Perl, Python, PHP, Tcl/Tk, MySQL or Linux) presented by the author of these notes, please see our public course schedule at

<http://www.wellho.net/course/index.html>

If you have a group of 4 or more trainees who require the same course at the same time, it will cost you less to have us run a private course for you. Please visit our onsite training page at

<http://www.wellho.net/course/otc.html>

which will give you details and costing information

1.4 Contact Details

Well House Consultants may be found online at

<http://www.wellho.net>

graham@wellho.net

technical contact

lisa@wellho.net

administration contact

Our full postal address is

404 The Spa

Melksham

Wiltshire

UK SN12 6QL

Phone +44 (0) 1225 708225

Fax +44 (0) 1225 707126

Putting the Java Language Together

You've learnt the fundamentals of Java – the language itself, the structure of classes and packages, and also the use of the utility classes in `java.util`. Now it's time to put all the fundamental elements together in a complete example. This module is written as an exercise, with a completed sample answer.

Sample Answer 4

The following is a complete exercise using the Java language and also the *java.io* and *java.util* APIs.

Sorting of letters is divided into "Areas" based around main sorting offices, and "Districts" which are more outlying places where post is sorted at the nearest main sorting office.

You are given a file containing over 1000 entries for areas and districts, and you are going to be required to write a series of applications to handle the data; here are some sample lines from the file.

ABERDEEN	Aberdeenshire	AB1,2	Aberdeen
Abbey Wood	London	SE2	London SE
Aberaeron	Dyfed	SA46	Swansea
Aberdare	Glamorgan	CF44	Cardiff

You can tell that somewhere is an "area" rather than a "district" if the name of the place (first column) is the same as the name of the sorting office (last column), and you'll also find that the Area name is in all capitals. The intermediate fields on each line are the county and postcode(s). The file also contains a number of blank lines.

Design and write a class or a series of classes to handle areas and districts, building them up in a collection object of some sort so that they can be searched based on the place name. Be aware that there some places share the same name (e.g. Newport), and you'll be required to look up the place by name and get ALL the results.

2.1 Sample Answer

```
#####
/home/trainee/coursework/PlaceFinder.java
```

A main program that reads a place name from the command line and reports on the place that has that name. This first example program is simplified so that it does NOT consider names that are applied to more than one place - see the later MultiFinder and MF2 example main classes for examples of the complete solution

```
#####
import java.io.*;
import java.util.*;
public class PlaceFinder {
public static void main(String [] args) {
    File Source = new File("postcodes");
    if (! Source.exists()) {
        System.err.println("No data file available");
    }
    BufferedReader Input=null;
    try {
        Input = new BufferedReader(
            new FileReader(Source));
    }
    catch (IOException e) {
        System.err.println("Cannot open file for read");
    }
    boolean havedata = true;
    Place current;
    Hashtable Lookup = new Hashtable();
    while (havedata) {
        try {
            String FromFile = Input.readLine();
            if (FromFile != null) {
                current = Place.make(FromFile);
            }
        }
    }
}
}
#####
```

```

        if (current != null) {
            String atplace = current.getName();
            Lookup.put(atplace,current);
        }
    } else {
        havedata=false;
    }
}
catch (IOException e) {
    System.err.println("Problem reading data");
    havedata=false;
}
}

current = (Place)(Lookup.get(args[0]));
System.out.println("Postcode is "+current.getPostcode());
System.out.println("Area is "+current.getArea());

}
}

```

```

#####
/home/trainee/coursework/Place.java

```

A Base abstract class for places; this contains most of the code for places, but is extended for "areas" and "districts" to take account of the different way those two types of place are handled. Note the static "make" method that takes a String that defines a place, works out what type of place it is, and constructs and returns either an area or a district

```

#####

```

```

public abstract class Place {

    String Name;
    String Postcode;
    String County;
    String Area;

    Place () {

    }

    public static Place make(String Incoming) {
        if (Incoming.length() < 61) return (null);

        String Name = (Incoming.substring(4,26)).trim();
        String County = (Incoming.substring(27,48)).trim();
        String Postcode = (Incoming.substring(48,61)).trim();
        String Area = (Incoming.substring(61)).trim();

        Place created;
        if (Name.equalsIgnoreCase(Area)) {
            created = new Area(Area,County,Postcode);
        } else {
            created = new District(Name,County,Postcode,Area);
        }
        return (created);
    }
}

```

```

public String getName() {
    return (Name);
}

public String getPostcode() {
    return (Postcode);
}

public String getCounty() {
    return (County);
}

public abstract String getArea();
}

```

```

#####
/home/trainee/coursework/PF1.java

```

Sample files "PF1" through "PF4" and "P3" through "P4" are simplified examples which show the application and class(es) under development.
 First experiment - reading data ...

```

#####
import java.io.*;
public class PF1 {
public static void main(String [] args) {
    File Source = new File("postcodes");
    if (! Source.exists()) {
        System.err.println("No data file available");
    }
    BufferedReader Input=null;
    try {
        Input = new BufferedReader(
            new FileReader(Source));
    }
    catch (IOException e) {
        System.err.println("Cannot open file for read");
    }
    try {
        String FromFile = Input.readLine();
        System.out.println(FromFile);
    }
    catch (IOException e) {
        System.err.println("No data in file");
    }
}
}

```

```

#####
/home/trainee/coursework/PF2.java

```

Second experiment - reading whole data file ...


```

P3 current;
Hashtable Lookup = new Hashtable();
while (havedata) {
    try {
        String FromFile = Input.readLine();
        if (FromFile != null) {
            current = P3.make(FromFile);
            if (current != null) {
                String atplace = current.getName();
                Lookup.put(atplace, current);
                System.out.println("Put " + atplace);
            }
        } else {
            havedata=false;
        }
    }
    catch (IOException e) {
        System.err.println("Problem reading data");
        havedata=false;
    }
}
}

#####
/home/trainee/coursework/P3.java
#####
public class P3 {

String Name;
String Postcode;
String County;
String Area;

P3 () {

}

public static P3 make(String Incoming) {
    if (Incoming.length() < 61) return (null);
    P3 created = new P3();
    created.Name = (Incoming.substring(4,26)).trim();
    return (created);
}

public String getName() {
    return (Name);
}

}

#####
/home/trainee/coursework/PF4.java

```

Fourth experiemnt - lookup on the data, ignoring "Area" v "District" differences

```
#####
import java.io.*;
import java.util.*;
public class PF4 {
public static void main(String [] args) {
    File Source = new File("postcodes");
    if (! Source.exists()) {
        System.err.println("No data file available");
    }
    BufferedReader Input=null;
    try {
        Input = new BufferedReader(
            new FileReader(Source));
    }
    catch (IOException e) {
        System.err.println("Cannot open file for read");
    }
    boolean havedata = true;
    P4 current;
    Hashtable Lookup = new Hashtable();
    while (havedata) {
        try {
            String FromFile = Input.readLine();
            if (FromFile != null) {
                current = P4.make(FromFile);
                if (current != null) {
                    String atplace = current.getName();
                    Lookup.put(atplace,current);
                }
            } else {
                havedata=false;
            }
        }
        catch (IOException e) {
            System.err.println("Problem reading data");
            havedata=false;
        }
    }

    current = (P4)(Lookup.get(args[0]));
    System.out.println("Postcode is "+current.getPostcode());

}
}
#####
/home/trainee/coursework/P4.java
#####
public class P4 {

String Name;
String Postcode;
String County;
String Area;
```

```

P4 () {
    }

public static P4 make(String Incoming) {
    if (Incoming.length() < 61) return (null);
    P4 created = new P4();
    created.Name = (Incoming.substring(4,26)).trim();
    created.County = (Incoming.substring(27,48)).trim();
    created.Postcode = (Incoming.substring(48,61)).trim();
    created.Area = (Incoming.substring(61)).trim();
    return (created);
}

public String getName() {
    return (Name);
}

public String getPostcode() {
    return (Postcode);
}

}

```

```

#####
/home/trainee/coursework/Area.java

```

Area subclass for the main applications; only the constructor and "getArea" methods vary in the subclasses
- everything else is inherited by both Areas and Districts.

```

#####
public class Area extends Place{

Area (String Name, String County, String Postcode) {
    this.Name = Name;
    this.Postcode = Postcode;
    this.County = County;
}

public String getArea () {
    return ("This is the Sorting town already");
}

}

```

```

#####
/home/trainee/coursework/District.java
#####
public class District extends Place{

```

```
String Area;
```

```
District (String Name, String County, String Postcode, String Area) {
    this.Name = Name;

```

```

        this.Postcode = Postcode;
        this.County = County;
        this.Area = Area;
    }

    public String getArea () {
        return (Area);
    }
}

```

```

#####
/home/trainee/coursework/MultiFinder.java

```

This example is a second application using the Place Area and District classes; It handles the data as a hash of Stacks so that it works if you have several places all of the same name.

```

#####
import java.io.*;
import java.util.*;
public class MultiFinder {
public static void main(String [] args) {
    File Source = new File("postcodes");
    if (! Source.exists()) {
        System.err.println("No data file available");
    }
    BufferedReader Input=null;
    try {
        Input = new BufferedReader(
            new FileReader(Source));
    }
    catch (IOException e) {
        System.err.println("Cannot open file for read");
    }
    boolean havedata = true;
    Place current;
    Hashtable Lookup = new Hashtable();
    while (havedata) {
        try {
            String FromFile = Input.readLine();
            if (FromFile != null) {
                current = Place.make(FromFile);
                if (current != null) {
                    String atplace = current.getName();

                    Stack Table = (Stack) Lookup.get(atplace);
                    if (Table == null) {
                        Table = new Stack();
                        Table.push(current);
                        Lookup.put(atplace,Table);
                    } else {
                        Table.push(current);
                    }
                }
            }
        } else {
            havedata=false;
        }
    }
}
}

```

```

        catch (IOException e) {
            System.err.println("Problem reading data");
            havedata=false;
        }
    }

    Stack report = (Stack)(Lookup.get(args[0]));
    while (! report.isEmpty()) {
        current = (Place) (report.pop());
        System.out.print(args[0]+" - "+current.getCounty());
        System.out.println(" ... postcode is "+current.getPostcode());
        System.out.println("Area is "+current.getArea());
    }
}
}
}

```

```

#####
/home/trainee/coursework/MF.java

```

A final example application; similar to MultiFinder, but the code has been rather more structured in the main application so that the individual elements of it can be easily re-used.

```

#####
import java.io.*;
import java.util.*;
public class MF {
public static void main(String [] args) {

    Hashtable info = read_data("postcodes");
    report_on(info,args[0]);

}

private static Hashtable read_data(String infile) {
    File Source = new File(infile);
    if (! Source.exists()) {
        System.err.println("No data file available");
    }
    BufferedReader Input=null;
    try {
        Input = new BufferedReader(
            new FileReader(Source));
    }
    catch (IOException e) {
        System.err.println("Cannot open file for read");
    }
    boolean havedata = true;
    Place current;
    Hashtable Lookup = new Hashtable();
    while (havedata) {
        try {
            String FromFile = Input.readLine();
            if (FromFile != null) {
                current = Place.make(FromFile);
                if (current != null) {

```

```

        String atplace = current.getName();

        Stack Table = (Stack) Lookup.get(atplace);
        if (Table == null) {
            Table = new Stack();
            Table.push(current);
            Lookup.put(atplace, Table);
        } else {
            Table.push(current);
        }
    }
} else {
    havedata=false;
}
}
catch (IOException e) {
    System.err.println("Problem reading data");
    havedata=false;
}
}
return (Lookup);
}

private static void report_on(Hashtable Lookup,String where) {
    Vector report = (Vector)(Lookup.get(where));
    for (int i=0;i<report.size();i++) {
        Place current = (Place) (report.elementAt(i));
        System.out.print(where+" - "+current.getCounty());
        System.out.print(" ... postcode is "+current.getPostcode());
        System.out.println(" ... Area is "+current.getArea());
    }
}
}
}

```

```

=====
postcodes
=====

```

ABERDEEN	Aberdeenshire	AB1,2	Aberdeen
Abbey Wood	London	SE2	London SE
Aberaeron	Dyfed	SA46	Swansea
Aberdare	Glamorgan	CF44	Cardiff
Aberdour	Fife	KY3	Kirkcaldy
Aberdovey	Gwynedd	LL35	Chester
Aberfeldy	Perthshire	PH15	Perth
Abergavenny	Gwent	NP7	Newport
Abergele	Clwyd	LL22	Chester
Aberlour	Banffshire	AB3	Aberdeen
Abertillery	Gwent	NP3	Newport
Aberystwyth	Dyfed	SY23	Shrewsbury
Abingdon	Oxfordshire	OX13,14	Oxford
Aboyne	Aberdeenshire	AB3	Aberdeen
Accrington	Lancashire	BB5	Preston
Acheracle	Angus	PA36	Paisley
Achnasheen	Ross	IV22	Inverness
Acton	London	W3	London W
Airdrie	Lanarkshire	ML6	Motherwell

Alcester	Warwickshire	B49,50	Birmingham
Aldeburgh	Suffolk	IP15	Ipswich
Alderley Edge	Cheshire	SK9	Stockport
Aldershot	Hampshire	GU11-13	Guildford

License

*These notes are distributed under the **Well House Consultants Open Training Notes License**. Basically, if you distribute it and use it for free, we'll let you have it for free. If you charge for its distribution of use, we'll charge.*

3.1 Open Training Notes License

Training notes distributed under the **Well House Consultants Open Training Notes License** (WHCOTNL) may be reproduced for any purpose PROVIDE THAT:

- This License statement is retained, unaltered (save for additions to the change log) and complete.
- No charge is made for the distribution, nor for the use or application thereof. This means that you can use them to run training sessions or as support material for those sessions, but you cannot then make a charge for those training sessions.
- Alterations to the content of the document are clearly marked as being such, and a log of amendments is added below this notice.
- These notes are provided "as is" with no warranty of fitness for purpose. Whilst every attempt has been made to ensure their accuracy, no liability can be accepted for any errors of the consequences thereof.

Copyright is retained by Well House Consultants Ltd, of 404, The Spa, Melksham, Wiltshire, UK, SN12 6QL - phone number +44 (1) 1225 708225. Email contact - Graham Ellis (graham@wellho.net).

Please send any amendments and corrections to these notes to the Copyright holder - under the spirit of the Open Distribution license, we will incorporate suitable changes into future releases for the use of the community.

If you are charged for this material, or for presentation of a course (Other than by Well House Consultants) using this material, please let us know. It is a violation of the license under which this notes are distributed for such a charge to be made, except by the Copyright Holder.

If you would like Well House Consultants to use this material to present a training course for your organisation, or if you wish to attend a public course is one is available, please contact us or see our web site - <http://www.wellho.net> - for further details.

Change log
Original Version, Well House Consultants, 2004

Updated by: _____ on _____
Updated by: _____ on _____
Updated by: _____ on _____
Updated by: _____ on _____
Updated by: _____ on _____
Updated by: _____ on _____

License Ends.