

More Sophisticated Behaviour

Technical Support System V3.0

Produced Dr. Siobhán Drohan
by: Mairead Meagher



Waterford Institute of Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Department of Computing and Mathematics
<http://www.wit.ie/>

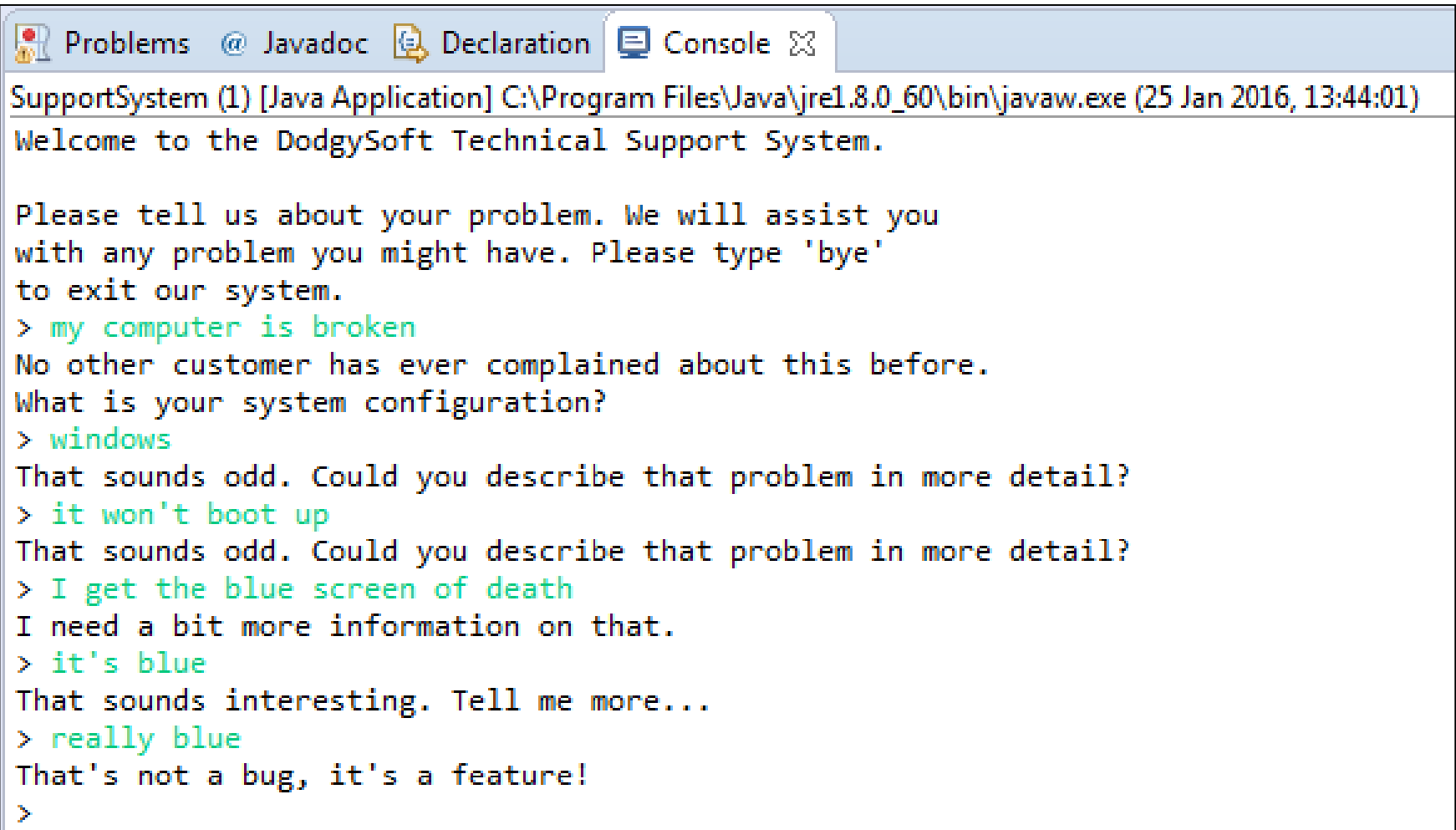
Topic List

- **Recap: Technical Support System V2**
- Technical Support System V3
 - Overview of the System.
 - Responder class: starting to generate a related response:
 - ArrayList
 - Map and HashMap
 - InputReader class: Tokenizing Strings:
 - Set and HashSet
 - Responder class: finishing the class.
 - SupportSystem class: a small change.

Technical Support System V2

- A console based, textual dialog system.
- In this version, the system provided a random response from a list of pre-defined responses e.g.:
 - "That sounds interesting. Tell me more..."
 - "I need a bit more information on that."
 - "Have you checked that you do not have a dll conflict?"
 - "That is explained in the manual. Have you read the manual?"
 - " That's not a bug, it's a feature!"
 - "Could you elaborate on that?"
 - etc.

Technical Support System V2

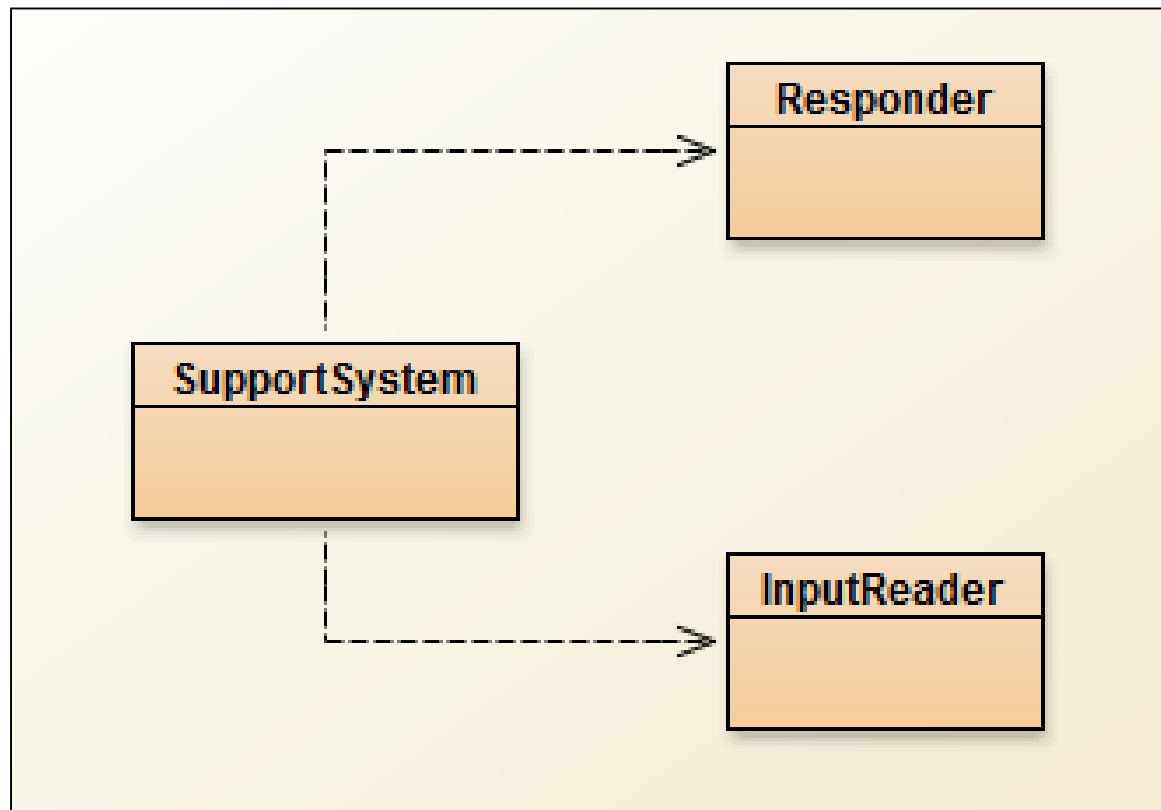


The screenshot shows a Java IDE window with a console tab selected. The console displays a chat conversation between a user and a technical support system. The user's input is shown in green text, and the system's responses are in black text. The chat starts with a welcome message, followed by the user reporting a broken computer. The system asks for more details, and the user mentions it won't boot up. The system asks for more details again, and the user reports a blue screen of death. The system asks for more information, and the user says it's blue. The system says that sounds interesting and asks for more details. The user says it's really blue. The system concludes that it's not a bug but a feature.

```
SupportSystem (1) [Java Application] C:\Program Files\Java\jre1.8.0_60\bin\javaw.exe (25 Jan 2016, 13:44:01)
Welcome to the DodgySoft Technical Support System.

Please tell us about your problem. We will assist you
with any problem you might have. Please type 'bye'
to exit our system.
> my computer is broken
No other customer has ever complained about this before.
What is your system configuration?
> windows
That sounds odd. Could you describe that problem in more detail?
> it won't boot up
That sounds odd. Could you describe that problem in more detail?
> I get the blue screen of death
I need a bit more information on that.
> it's blue
That sounds interesting. Tell me more...
> really blue
That's not a bug, it's a feature!
>
```

Class Diagram V2



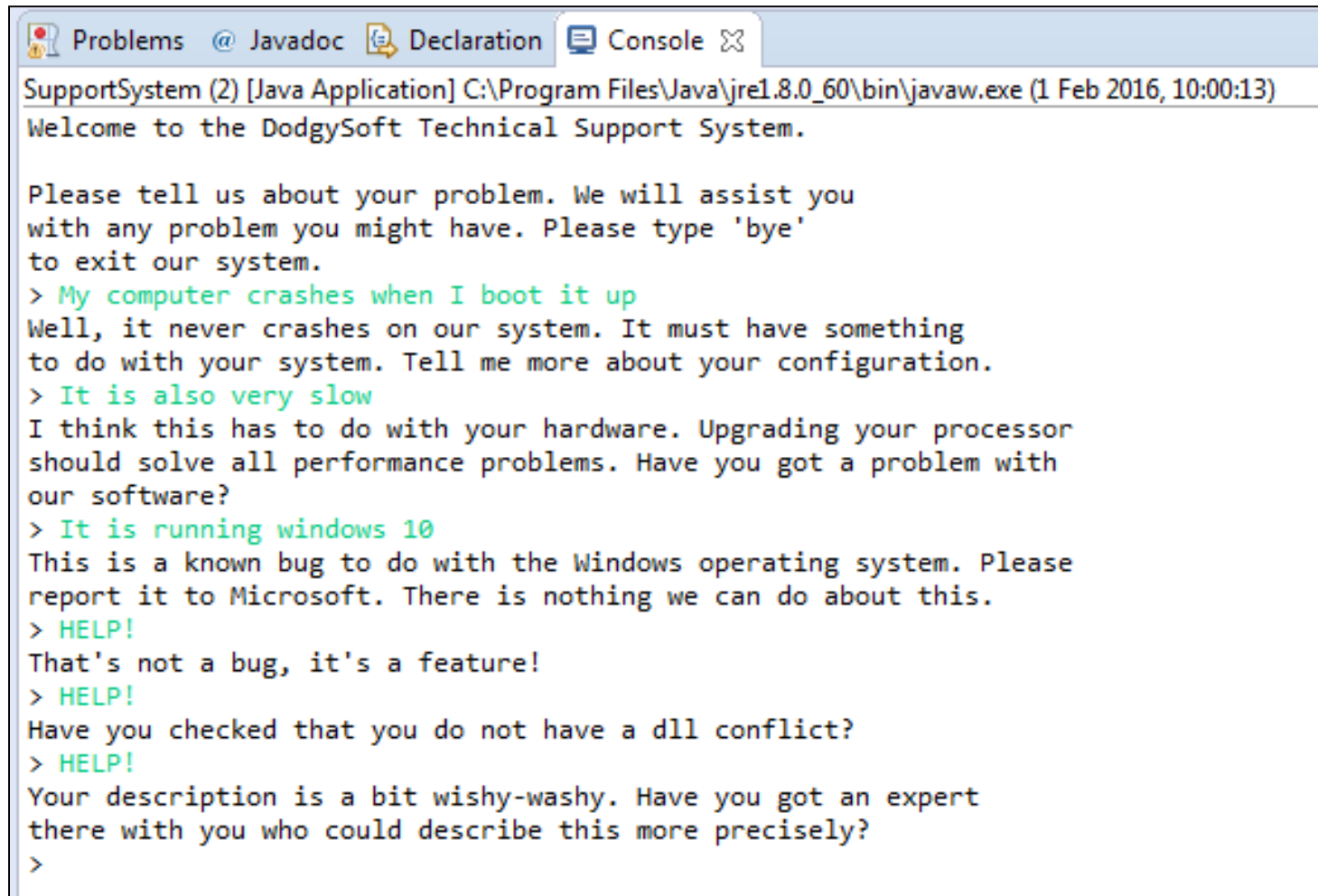
Topic List

- Recap: Technical Support System V2
- Technical Support System V3
 - Overview of the System.
 - Responder class: starting to generate a related response:
 - ArrayList
 - Map and HashMap
 - InputReader class: Tokenizing Strings:
 - Set and HashSet
 - Responder class: finishing the class.
 - SupportSystem class: a small change.

Technical Support System V3

- A console based, textual dialog system.
- Based on the user input, the system provides a context-sensitive, generated response from a list of pre-defined responses. If the system cannot find a suitable generated response, it returns a random one.

Technical Support System V3



The screenshot shows a Java IDE window with a tab labeled 'Console'. The console output displays a Java application titled 'SupportSystem (2) [Java Application]' running at 'C:\Program Files\Java\jre1.8.0_60\bin\javaw.exe' on '1 Feb 2016, 10:00:13'. The application is a 'DodgySoft Technical Support System' that prompts the user for a problem. The user provides three inputs: 'My computer crashes when I boot it up', 'It is also very slow', and 'It is running windows 10'. The system responds with advice and eventually asks for a more precise description of the problem.

```
SupportSystem (2) [Java Application] C:\Program Files\Java\jre1.8.0_60\bin\javaw.exe (1 Feb 2016, 10:00:13)
Welcome to the DodgySoft Technical Support System.

Please tell us about your problem. We will assist you
with any problem you might have. Please type 'bye'
to exit our system.
> My computer crashes when I boot it up
Well, it never crashes on our system. It must have something
to do with your system. Tell me more about your configuration.
> It is also very slow
I think this has to do with your hardware. Upgrading your processor
should solve all performance problems. Have you got a problem with
our software?
> It is running windows 10
This is a known bug to do with the Windows operating system. Please
report it to Microsoft. There is nothing we can do about this.
> HELP!
That's not a bug, it's a feature!
> HELP!
Have you checked that you do not have a dll conflict?
> HELP!
Your description is a bit wishy-washy. Have you got an expert
there with you who could describe this more precisely?
>
```


Topic List

- Recap: Technical Support System V2
- Technical Support System V3
 - Overview of the System.
 - Responder class: starting to generate a related response:
 - ArrayList
 - Map and HashMap
 - InputReader class: Tokenizing Strings:
 - Set and HashSet
 - Responder class: finishing the class.
 - SupportSystem class: a small change.

How do we influence the generated response?

- What if we had a set of words that are likely to occur in a typical question?
- What if we then associated these words with particular responses?
- Then, if the input from the user contains one of our known words, generate a related response!

ArrayList

- Can we use an **ArrayList** for this purpose?
- Will it let us store “key=value” pairs?
- No! We need a different data structure.
- A **Map** will store “key=value” pairs though!

Collection Interface Concrete Implementation Classes

Class	Map	Set	List	Ordered	Sorted
HashMap	X			No	No
Hashtable	X			No	No
TreeMap	X			Sorted	By natural order or custom comparison rules
LinkedHashMap	X			By insertion order or last access order	NO
HashSet		X		No	No
TreeSet		X		Sorted	By natural order or custom comparison rules
LinkedHashSet		X		By insertion order	No
ArrayList			X	By index	No
Vector			X	By index	No
LinkedList			X	By index	No
PriorityQueue				Sorted	By to-do order

Topic List

- Recap: Technical Support System V2
- Technical Support System V3
 - Overview of the System.
 - Responder class: starting to generate a related response:
 - ArrayList
 - Map and HashMap
 - InputReader class: Tokenizing Strings:
 - Set and HashSet
 - Responder class: finishing the class.
 - SupportSystem class: a small change.

Maps

- Maps are collections that contain pairs of values.
- Pairs consist of a key and a value.
- Lookup works by supplying a key, and retrieving a value.
- An example: a telephone book...use the name to look up a phone number.

Using maps

- A map with Strings as keys and values.

:HashMap

"Charles Nguyen"

"(531) 9392 4587"

"Lisa Jones"

"(402) 4536 4674"

"William H. Smith"

"(998) 5488 0123"

ArrayList Vs Map

- In an ArrayList each entry stores one object whereas in a Map, each entry has a pair of objects (key=value).
- In ArrayList, you use an integer index to look up the object, whereas in a Map, you use the key object to look up the value object.

More on Map

- Looking up a value associated with a key is easy!
- However, reverse lookup (finding a key for a value) is not so easy.
- Ideal for one-way lookup using the key.
- A map cannot contain duplicate keys; each key can map to at most one value.
- We will use the **HashMap** class.

HashMap Methods

java.util

Class HashMap<K,V>

Method Summary

Methods

Modifier and Type	Method and Description
void	clear() Removes all of the mappings from this map.
Object	clone() Returns a shallow copy of this HashMap instance: the keys and values themselves are not cloned.
boolean	containsKey(Object key) Returns true if this map contains a mapping for the specified key.
boolean	containsValue(Object value) Returns true if this map maps one or more keys to the specified value.
Set<Map.Entry<K,V>>	entrySet() Returns a Set view of the mappings contained in this map.
V	get(Object key) Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.
boolean	isEmpty() Returns true if this map contains no key-value mappings.
Set<K>	keySet() Returns a Set view of the keys contained in this map.
V	put(K key, V value) Associates the specified value with the specified key in this map.
void	putAll(Map<? extends K,? extends V> m) Copies all of the mappings from the specified map to this map.
V	remove(Object key) Removes the mapping for the specified key from this map if present.
int	size() Returns the number of key-value mappings in this map.
Collection<V>	values() Returns a Collection view of the values contained in this map.

Using HashMap

```
HashMap <String, String> phoneBook = new HashMap<String, String>();  
  
phoneBook.put("Charles Nguyen", "(531) 9392 4587");  
phoneBook.put("Lisa Jones", "(402) 4536 4674");  
phoneBook.put("William H. Smith", "(998) 5488 0123");  
  
String phoneNumber = phoneBook.get("Lisa Jones");  
System.out.println(phoneNumber);
```

HashMap in Tech Support System V3

In the Responder class, we will now use HashMap to store “Key-Value” pairs for context-sensitive responses e.g.

Key	Value
windows	This is a known bug to do with the Windows operating system. Please report it to Microsoft. There is nothing we can do about this.
slow	I think this has to do with your hardware. Upgrading your processor should solve all performance problems. Have you got a problem with our software?
bug	Well, you know, all software has some bugs. But our software engineers are working very hard to fix them. Can you describe the problem a bit further?
performance	Performance was quite adequate in all our tests. Are you running any other processes in the background?

HashMap in Tech Support System V3

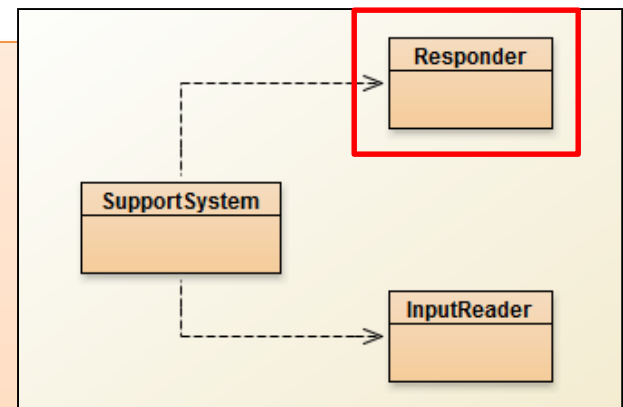
```
responseMap.put("crashes", "Well, it never crashes on our  
system. It must have something\n" + "to do with your system.  
Tell me more about your configuration.");
```

- Whenever someone enters the word “crashes”, we can look up and print the attached response.
- Lets look at the HashMap code in the Responder class!

```
import java.util.HashMap;
import java.util.ArrayList;
import java.util.Random;

public class Responder
{
    // Used to map key words to responses.
    private HashMap<String, String> responseMap;
    // Default responses to use if we don't recognise a word.
    private ArrayList<String> defaultResponses;
    private Random randomGenerator;

    public Responder()
    {
        responseMap = new HashMap<String, String>();
        fillResponseMap();
        defaultResponses = new ArrayList<String>();
        fillDefaultResponses();
        randomGenerator = new Random();
    }
}
```



V3.0 Responder
changes (in red)

V3.0 Responder changes (in red)

```
private void fillDefaultResponses() {  
    defaultResponses.add("That sounds odd. Could you describe that problem in more detail?");  
    defaultResponses.add("No other customer has ever complained about this before. \n" +  
        "What is your system configuration?");  
    defaultResponses.add("That sounds interesting. Tell me more...");  
    defaultResponses.add("I need a bit more information on that.");  
    defaultResponses.add("Have you checked that you do not have a dll conflict?");  
    defaultResponses.add("That is explained in the manual. Have you read the manual?");  
    defaultResponses.add("Your description is a bit wishy-washy. Have you got an expert\n" +  
        "there with you who could describe this more precisely?");  
    defaultResponses.add("That's not a bug, it's a feature!");  
    defaultResponses.add("Could you elaborate on that?");  
}
```

```
private String pickDefaultResponse()  
{  
    // Pick a random number for the index in the default response list.  
    // The number will be between 0 (inclusive) and the size of the list (exclusive).  
    int index = randomGenerator.nextInt(defaultResponses.size());  
    return defaultResponses.get(index);  
}
```

V3.0 Responder changes (in red)

```
private void fillResponseMap()
{
    responseMap.put("crash",
        "Well, it never crashes on our system. It must have something\n" +
        "to do with your system. Tell me more about your configuration.");
    responseMap.put("crashes",
        "Well, it never crashes on our system. It must have something\n" +
        "to do with your system. Tell me more about your configuration.");
    responseMap.put("slow",
        "I think this has to do with your hardware. Upgrading your processor\n" +
        "should solve all performance problems. Have you got a problem with\n" +
        "our software?");
    responseMap.put("performance",
        "Performance was quite adequate in all our tests. Are you running\n" +
        "any other processes in the background?");
    responseMap.put("bug",
        "Well, you know, all software has some bugs. But our software engineers\n" +
        "are working very hard to fix them. Can you describe the problem a bit\n" +
        "further?");
    responseMap.put("buggy",
        "Well, you know, all software has some bugs. But our software engineers\n" +
        "are working very hard to fix them. Can you describe the problem a bit\n" +
        "further?");
    responseMap.put("windows",
        "This is a known bug to do with the Windows operating system. Please\n" +
        "report it to Microsoft. There is nothing we can do about this.");
    // and so on...
}
```


Topic List

- Recap: Technical Support System V2
- Technical Support System V3
 - Overview of the System.
 - Responder class: starting to generate a related response:
 - ArrayList
 - Map and HashMap
 - InputReader class: Tokenizing Strings:
 - Set and HashSet
 - Responder class: finishing the class.
 - SupportSystem class: a small change.

Tokenizing Strings

- We have a HashMap containing a series words with appropriate responses.
- Now we need to search the String of words the user entered on the console to see if they typed in any of the words stored in the HashMap.
- We need to “split” the String of words entered by the user into individual words and store them in a collection (e.g. Array) →Tokenizing Strings.
- We need a new data structure for this type of data.

Set

- A **Set** is a collection that stores each individual element at most once (i.e. unique elements).
- It does not maintain any specific order.
- The coding for **Set** is very similar to **ArrayList** coding.

Using sets

```
import java.util.HashSet;
import java.util.Iterator;
...
HashSet<String> mySet = new HashSet<String>();
```

```
mySet.add("one");
mySet.add("two");
mySet.add("three");
```

Compare this
to ArrayList
code!

```
Iterator<String> it = mySet.iterator();
while(it.hasNext()) {
    call it.next() to get the next object
    do something with that object
}
```

What is the Difference between Set and List?

List (e.g. ArrayList):

- keeps all elements entered in the desired order,
- provides access to elements by index
- can contain the same element multiple times.

Set (e.g. HashSet):

- does not maintain any specific order
- ensures each element is in the set at most once (entering an element a second time has no effect).

Returning to Tokenizing Strings

InputReader class

```
//V2 Code
```

```
import java.util.Scanner;
```

```
public class InputReader{
```

```
    Scanner input;
```

```
    public InputReader(){
```

```
        input = new Scanner(System.in);
```

```
    }
```

```
    /**
```

```
     * Read a line of text from standard input (the text terminal),
```

```
     * and return it as a String.
```

```
     *
```

```
     * @return A String typed by the user.
```

```
     */
```

```
    public String getInput() {
```

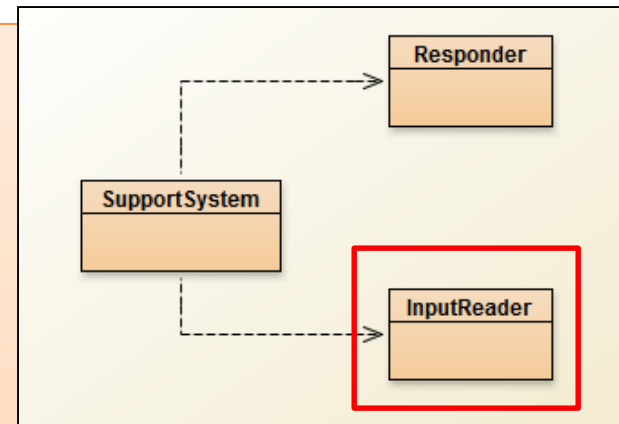
```
        System.out.print("> ");           // print prompt
```

```
        String inputLine = input.nextLine().trim().toLowerCase();
```

```
        return inputLine;
```

```
    }
```

```
}
```



This class will be changed to split the input into a primitive array of Strings.

//V3 Code

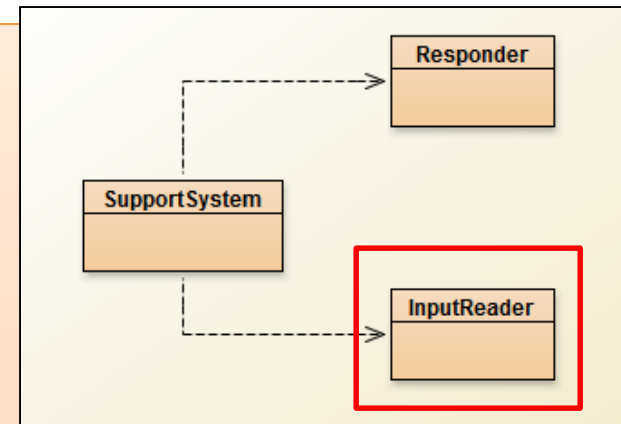
```
import java.util.Scanner;
public class InputReader{
    Scanner input;

    public InputReader(){
        input = new Scanner(System.in);
    }

    public HashSet<String> getInput()
    {
        System.out.print("> ");           // print prompt
        String inputLine = input.nextLine().trim().toLowerCase();

        String[] wordArray = inputLine.split(" "); // split at spaces

        // add words from array into hashset
        HashSet<String> words = new HashSet<String>();
        for(String word : wordArray) {
            words.add(word);
        }
        return words;
    }
}
```



Changes for
V3

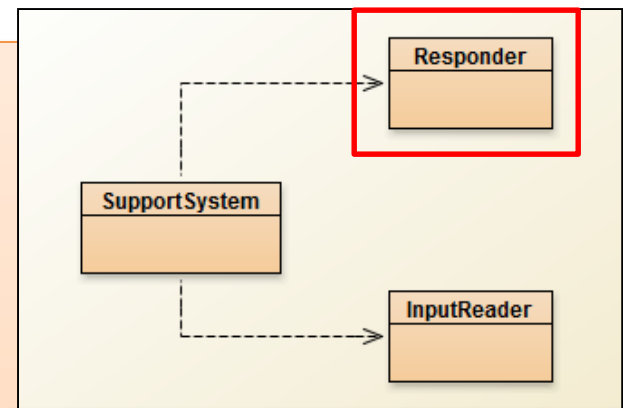
Topic List

- Recap: Technical Support System V2
- Technical Support System V3
 - Overview of the System.
 - Responder class: starting to generate a related response:
 - ArrayList
 - Map and HashMap
 - InputReader class: Tokenizing Strings:
 - Set and HashSet
 - Responder class: finishing the class.
 - SupportSystem class: a small change.

```
import java.util.HashMap;
import java.util.HashSet;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Random;
```

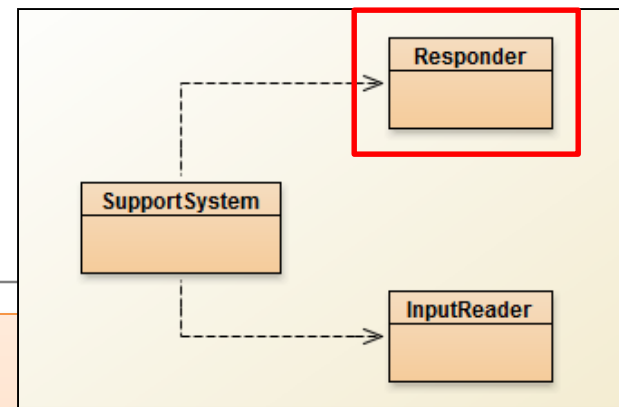
```
public class Responder
{
    // Used to map key words to responses.
    private HashMap<String, String> responseMap;
    // Default responses to use if we don't recognise a word.
    private ArrayList<String> defaultResponses;
    private Random randomGenerator;

    public Responder()
    {
        responseMap = new HashMap<String, String>();
        fillResponseMap();
        defaultResponses = new ArrayList<String>();
        fillDefaultResponses();
        randomGenerator = new Random();
    }
}
```



V3.0 Responder
MORE changes (in red)
to handle a HashSet of
Strings passed into the
generateResponse
method.

```
public String generateResponse(HashSet<String> words)
{
    Iterator<String> it = words.iterator();
    while(it.hasNext()) {
        String word = it.next();
        String response = responseMap.get(word);
        if(response != null) {
            return response;
        }
    }
    // If we get here, none of the words from the input line were recognized.
    // In this case we pick one of our default responses (what we say when
    // we cannot think of anything else to say...)
    return pickDefaultResponse();
}
```



V3.0 Responder
MORE changes (in red)
to handle a HashSet of
Strings passed into the
generateResponse
method.

Topic List

- Recap: Technical Support System V2
- Technical Support System V3
 - Overview of the System.
 - Responder class: starting to generate a related response:
 - ArrayList
 - Map and HashMap
 - InputReader class: Tokenizing Strings:
 - Set and HashSet
 - Responder class: finishing the class.
 - SupportSystem class: a small change.

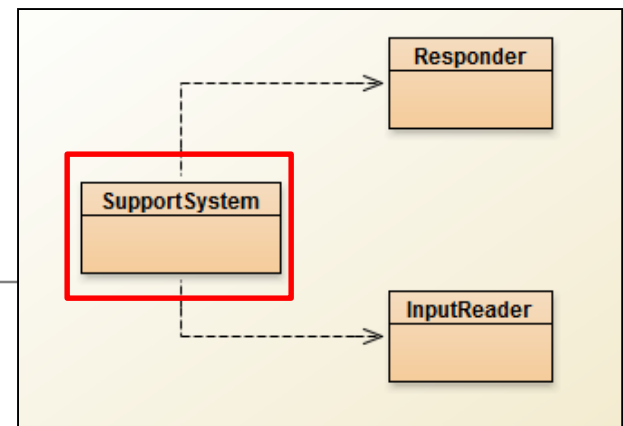
//V2 code

```
public class SupportSystem
{
    private InputReader reader;
    private Responder responder;

    public SupportSystem() {
        reader = new InputReader();
        responder = new Responder();
    }

    public static void main(String[] args){
        SupportSystem app = new SupportSystem();
        app.start();
    }

    public void start(){
        printWelcome();
        String input = reader.getInput();
        while(! input.startsWith("bye")) {
            String response = responder.generateResponse();
            System.out.println(response);
            input = reader.getInput();
        }
        printGoodbye();
    }
}
```



Slight change
will be made in
this class,
mainly in the
start() method.

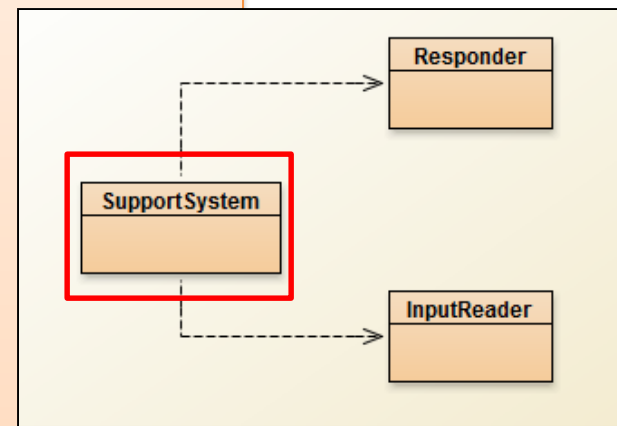
//V3 code

```
public class SupportSystem
{
    private InputReader reader;
    private Responder responder;

    public SupportSystem() {
        reader = new InputReader();
        responder = new Responder();
    }

    public static void main(String[] args){
        SupportSystem app = new SupportSystem();
        app.startSupport();
    }

    public void startSupport(){
        printWelcome();
        HashSet<String> input = reader.getInput();
        while(!input.contains("bye")) {
            String response = responder.generateResponse(input);
            System.out.println(response);
            input = reader.getInput();
        }
        printGoodbye();
    }
}
```



Use the
HashSet
and add the
relevant
import
statements
at the top!

**Any
Questions?**

