

5 More sophisticated behavior

Using library classes to implement some more advanced functionality

Main concepts to be covered

- Maps and sets
- Random number generation
- *Equality vs identity*
- *Class variables*
- *Constants*
- *Information hiding*

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 5 2

A Technical Support System

- A textual dialog system
- Idea based on 'Eliza' by Joseph Weizenbaum (MIT, 1960s)
- *Explore the online code!*

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 5 3

Main loop structure

```
boolean finished = false;

while (!finished) {

    do something

    if (exit condition) {
        finished = true;
    }
    else {
        do something more
    }
}
```

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 5 4

Main loop body

```
String input = reader.getInput();
...
String response = responder.generateResponse();
System.out.println(response);
```

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 5 5

The exit condition

```
String input = reader.getInput();

if (input.startsWith("bye")) {
    finished = true;
}
```

- Look for 'startsWith' in `java.lang.String`

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 5 6

Random number generator

- The library class `Random` can be used to generate (pseudo) random numbers

```
import java.util.Random;
...
Random randomGenerator = new Random();
...
int index1 = randomGenerator.nextInt();
int index2 = randomGenerator.nextInt(100);
```

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 5 7

Generating random responses

```
public Responder()
{
    randomGenerator = new Random();
    responses = new ArrayList<String>();
    fillResponses();
}

public String generateResponse()
{
    int index = randomGenerator.nextInt(responses.size());
    return responses.get(index);
}

public void fillResponses()
{
    ...
}
```

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 5 8

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.

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 5 9

Using maps

- A map with Strings as keys and values

:HashMap	
"Uno Holmer"	"+46(0)317725730"
"Christer Carlsson"	"+46(0)317721038"
"Javaakuten"	"999999"

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 5 10

Using maps

```
HashMap<String, String> phoneBook =
    new HashMap<String, String>();

phoneBook.put("Uno Holmer", "+46(0)317725730");
phoneBook.put("C. Carlsson", "+46(0)317721038");
phoneBook.put("Javaakuten", "999999");

String phoneNumber = phoneBook.get("Javaakuten");
System.out.println(phoneNumber);
```

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 5 11

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");

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

Compare this
to ArrayList
code!

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 5 12

Sets contain no duplicates

Example.

```
String[] words =
{ "aaa", "bb", "aaa", "ccc", "d", "aaa", "bb", "e", "ccc" };

HashSet<String> set = new HashSet<String>();
for (String word : words)
    set.add(word);

for (String s : set)
    System.out.println(s);
```

program output

```
bb
aaa
ccc
e
d
```

in arbitrary order (HashSet)

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 5 13

Generating “smart” responses

```
private HashMap<String,String> responseMap;
...
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;
    }
    return pickDefaultResponse();
}
```

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 5 14

... using the for-each loop

```
private HashMap<String,String> responseMap;
...
public String generateResponse(HashSet<String> words)
{
    for (String word : words) {
        String response = responseMap.get(word);
        if(response != null)
            return response;
    }
    return pickDefaultResponse();
}
```

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 5 15

Tokenizing a string (Splitting a string into words)

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

    String[] wordArray = inputLine.split();
    HashSet<String> words = new HashSet<String>();

    for (String word : wordArray)
        words.add(word);

    return words;
}
```

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 5 16

Some further topics

- Equality vs identity
- Class variables
- Constants
- Information hiding

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 5 17

Side note: String equality

```
if(input == "bye") {
```

tests identity

```
    ...
}
```

```
if(input.equals("bye")) {
```

tests equality

```
    ...
}
```

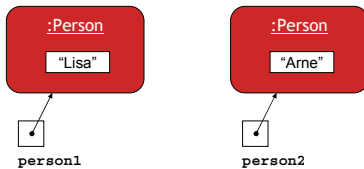
- Strings should always be compared with `.equals`

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 5 18

Identity vs equality 1

Other (non-String) objects:



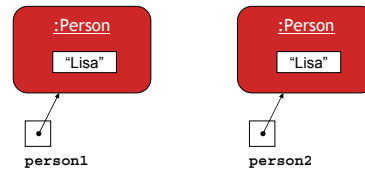
`person1 == person2 ?`

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 5 19

Identity vs equality 2

Other (non-String) objects:



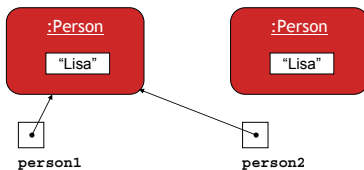
`person1 == person2 ?`

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 5 20

Identity vs equality 3

Other (non-String) objects:



`person1 == person2 ?`

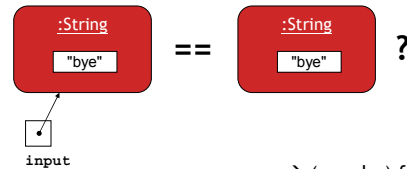
Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 5 21

Identity vs equality (Strings)

```
String input = reader.getInput();
if(input == "bye") {
    ...
}
```

== tests identity



→ (may be) false!

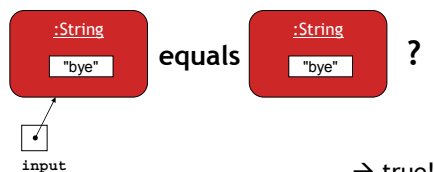
Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 5 22

Identity vs equality (Strings)

```
String input = reader.getInput();
if(input.equals("bye")) {
    ...
}
```

equals tests equality

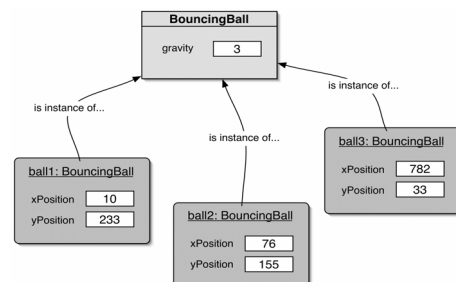


→ true!

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 5 23

Class variables



Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 5 24

Constants

```
private static final int gravity = 3;
```

- **private**: access modifier, as usual
- **static**: class variable
- **final**: constant

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 5 25

Public vs private

- Public attributes (instance variables, constructors, methods) are accessible to other classes.
- Instance variables should not be public.
- Private attributes are accessible only within the same class.
- Only methods that are intended for other classes should be public.

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 5 26

Information hiding

- Data belonging to one object is hidden from other objects.
- Know what an object can do, not how it does it.
- Information hiding increases the level of *independence*.
- Independence of modules is important for large systems and maintenance.

Object oriented programming, DAT042, D2, 11/12, lp 1

Lecture 5 27