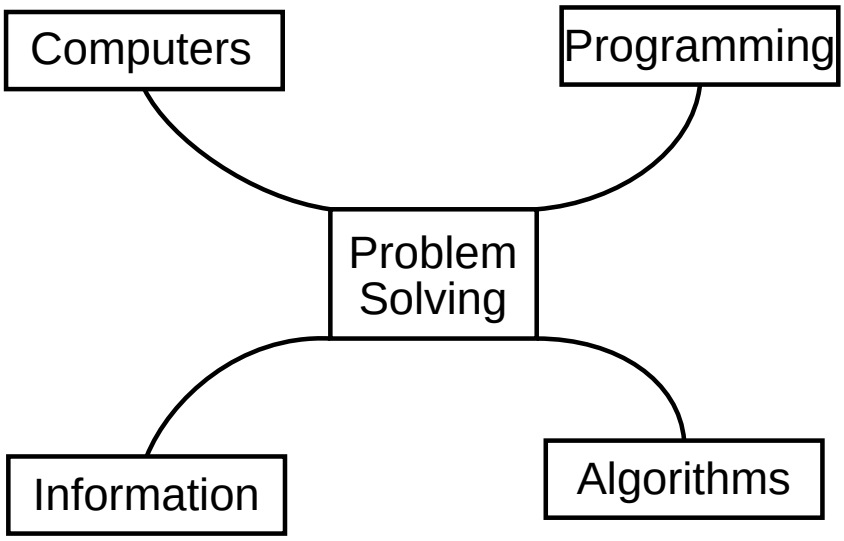
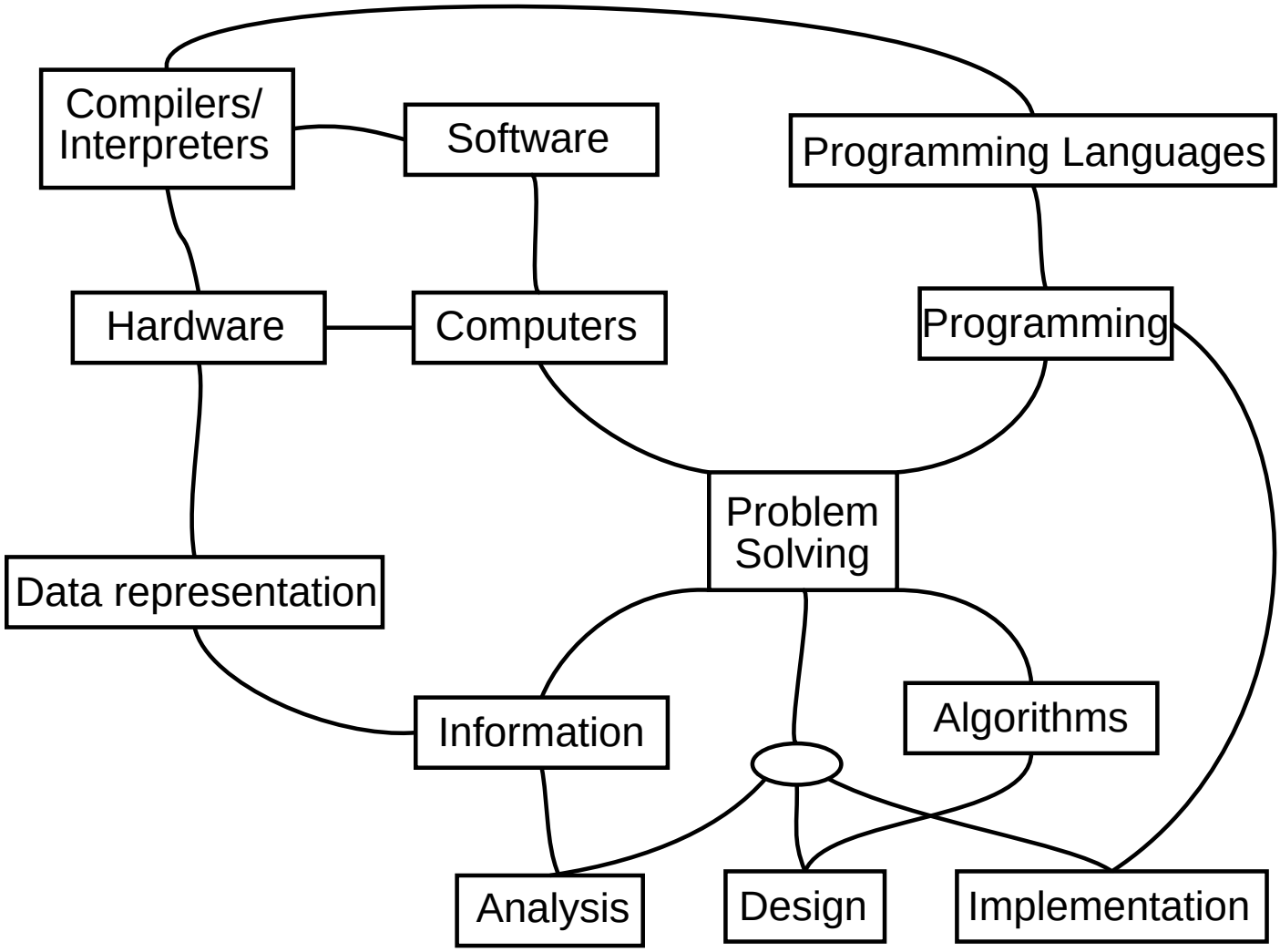
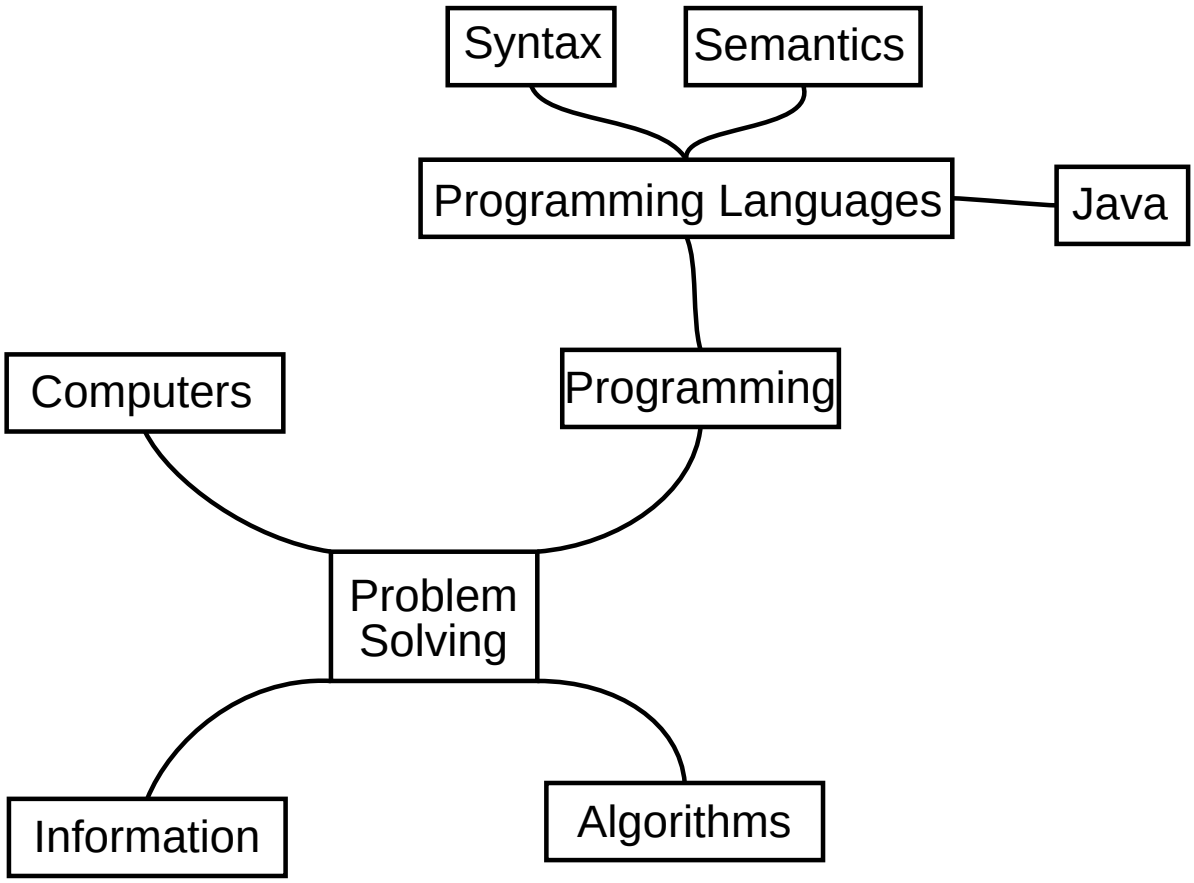
Road map


Computers
Programming
Problem Solving
Information
Algorithms

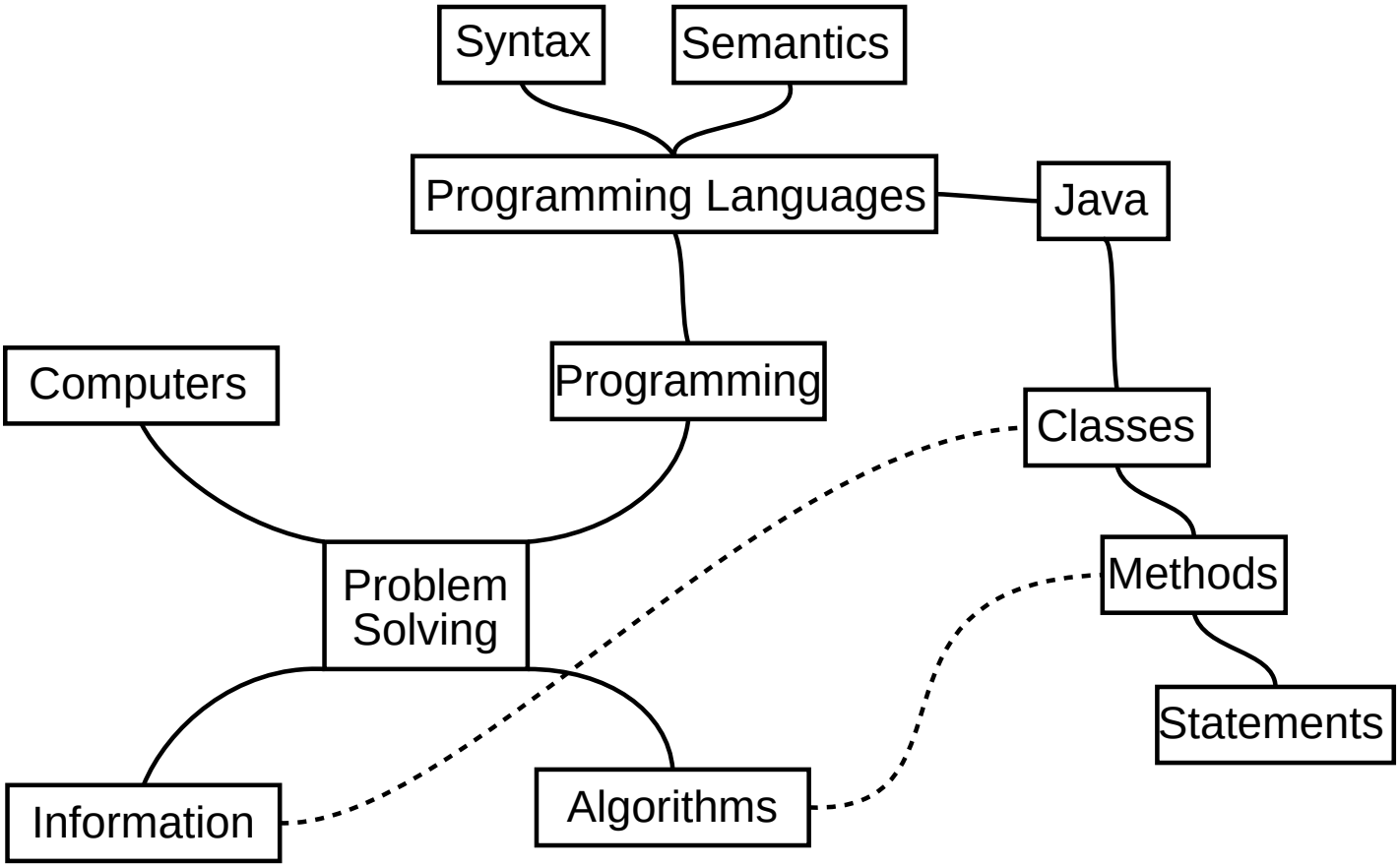
Road map


Compilers/
Interpreters
Software
Programming Languages
Hardware
Computers
Programming
Problem
Solving
Data representation
Information
Algorithms
Analysis
Design
Implementation

Road map


Syntax
Semantics
Programming Languages
Java
Computers
Programming
Problem Solving
Information
Algorithms

Road map


Syntax
Semantics
Programming Languages
Java
Computers
Programming
Classes
Problem Solving
Methods
Statements
Information
Algorithms

Statements

- Variable declaration

```
type identifier;
```

- Assignment

```
variable = expression;
```

- User Interface: output

```
System.out.println(string_expression);
```

- User Interface: input

```
variable = Keyboard.readType();
```

Primitive Data Types

General category	Type	Description	Examples
Numeric	int	Integers	0,1,-3
	long	Long integers	65537l
	short	Short integers	2,-6
	byte	Bytes	255
	float	Rationals	1.33f
	double	Rationals	1.618
Text	char	Single characters	'x', ' '
	String	Sequences of characters	"abc"
Logic	boolean	Truth values	true, false

Data conversion

- Sometimes it is useful to look at data as if they were from a different type
- For example:
  - Adding an integer and a double
  - Obtaining the ASCII code of a character
- Forms of data conversion:
  - Implicit:
    - Assignment conversion
    - Promotion
  - Explicit: Casting

Data conversion

- Assignment conversion: A value of one type is assigned to a variable of a different type, as long as the types are compatible

```
int n = 7;
double d = n;
long k = n;
int m = d; // Wrong: compile-time error
```

- Promotion: an expression "promotes" the types of its operands to its "largest" type

```
int m = 8;
float x = 3.0f, y;
y = x + m;
```

Data conversion

- Casting expressions (not a statement)

  (*type*)*expression*

- Examples:

```
int n = 3;
double p;
p = (double)n + 4.0;

int a = 3, b = 8;
float c, d;
c = b/a;
d = (float)b/a;
System.out.println(c);  // 2.0
System.out.println(d);  // 2.666666...
```

Data conversion

```
double r = 2.41;
int a;
a = r; // Error
```

Data conversion

```
double r = 2.41;
int a;
a = (int)r;    //OK:  Narrowing casting
```

Data conversion

- There are two types of casting:
  - Narrowing conversions: from a type which requires more memory to a type that requires less
  - Widening conversions: from a type which requires less memory to a type which requires more

- If `expression` has type `t`, and `t` requires more memory than type `s`, then `(s)expression` is a narrowing conversion (e.g. `int` to `byte`, `double` to `float`, `float` to `int`, ...)

- If `expression` has type `t`, and `t` requires less memory than type `s`, then `(s)expression` is a widening conversion (e.g. `byte` to `double`, `long` to `int`, ...)

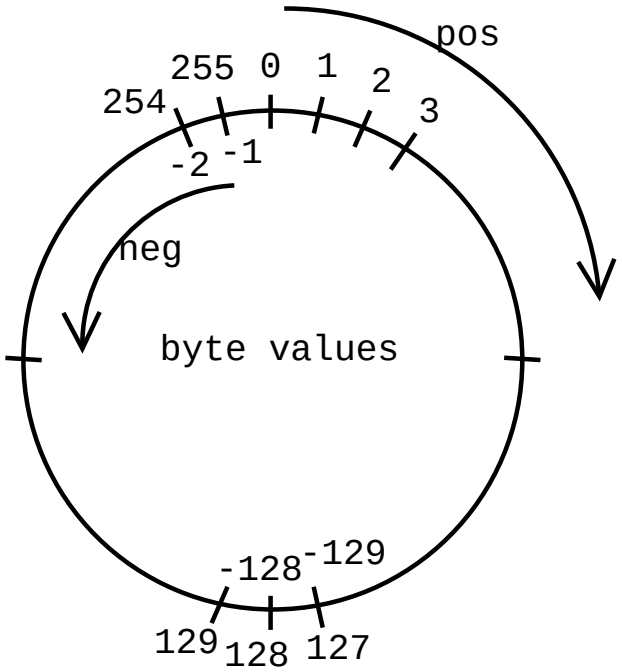
Data conversion

- Widening conversions are safe: no loss of information
- Narrowing conversions are not safe: possible loss of information

```
float x = 2.71f;
int i = (int)x;
// i == 2

int k = 130;
byte b = (byte)k;
// b = -126
```

Data conversion



128 = -128	byte b	b+$k2^8$=b
129 = -127	int i	i+$k2^{32}$=i
256 = 0	k is any integer	
257 = 1		

Data conversion

- Problem: Compute the average number of courses taken by three different students

- Analysis:
  - Information involved: number of courses taken by each student, average
  - Input: three (positive) integers n_1, n_2, and n_3
  - Output: the *average*

- Design:

  1. Obtain input, n_1, n_2, and n_3
  2. Compute average according to

$$average = \frac{n_1 + n_2 + n_3}{3}$$

  3. Print *average* as output

Data conversion

- Implementation (first attempt):

```
import cs1.Keyboard;
public class ComputeAverage {
  public static void main(String[] args)
  {
    int n1, n2, n3;
    float average;

    System.out.print("Enter n1: ");
    n1 = Keyboard.readInt();
    System.out.print("Enter n2: ");
    n2 = Keyboard.readInt();
    System.out.print("Enter n3: ");
    n3 = Keyboard.readInt();

    average = (n1 + n2 + n3) / 3;

    System.out.println("Average="+average);
  }
}
```

Data conversion

- Implementation (first attempt):

```
import cs1.Keyboard;
public class ComputeAverage {
  public static void main(String[] args)
  {
    int n1, n2, n3;
    float average;

    System.out.print("Enter n1: ");
    n1 = Keyboard.readInt();
    System.out.print("Enter n2: ");
    n2 = Keyboard.readInt();
    System.out.print("Enter n3: ");
    n3 = Keyboard.readInt();

    average = (n1 + n2 + n3) / 3;

    System.out.println("Average="+average);
  }
}
```

Data conversion

- Implementation (second attempt):

```
import cs1.Keyboard;
public class ComputeAverage {
  public static void main(String[] args)
  {
    int n1, n2, n3;
    float average;

    System.out.print("Enter n1: ");
    n1 = Keyboard.readInt();
    System.out.print("Enter n2: ");
    n2 = Keyboard.readInt();
    System.out.print("Enter n3: ");
    n3 = Keyboard.readInt();

    average = (float)(n1 + n2 + n3) / 3;

    System.out.println("Average="+average);
  }
}
```

Precedence

Precedence	Operator	Operation	Associativity
1	+	Unary plus	right to left
	-	Unary minus	
	!	Logical negation (NOT)	
2	(*type*)	Type cast	right to left
3	*	Multiplication	left to right
	/	Division	
	%	Remainder (modulo)	
4	+	Addition	left to right
	-	Substraction	
	+	String concatenation	
5	<	Less than	left to right
	<=	Less than or equal to	
	>	Greater than	
	>=	Greater than or equal to	
6	==	Equals to	left to right
	!=	Different to	
7	&&	Logical conjunction (AND)	left to right
8	||	Logical disjunction (OR)	left to right

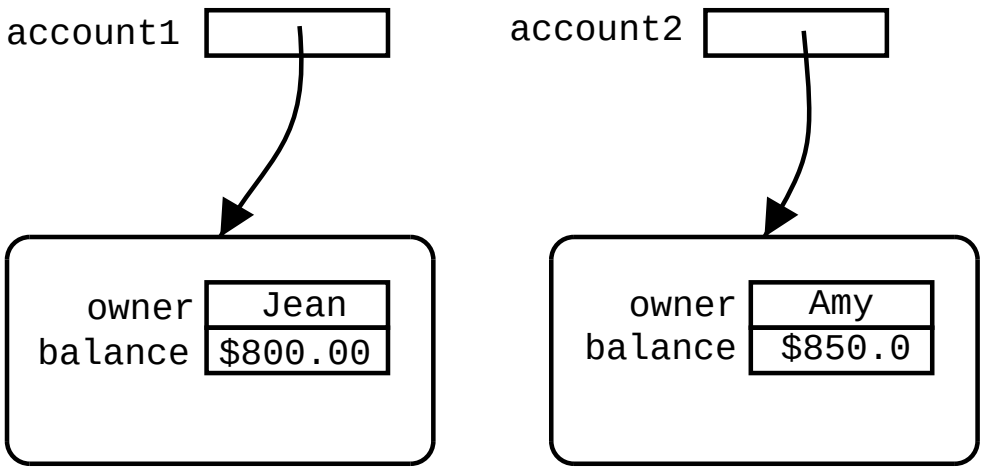
Objects and Classes

- Information in a Java program is represented by either
  - Primitive data (e.g. numbers, booleans)
  - Objects (composite data)
- Primitive data is defined by primitive data types (int, char, boolean)
- Objects are defined by Classes: the type of an object is a class
- Classes are given by a list of methods
- Methods: operations that can be performed on objects of the class where the method is defined)

Objects and Classes

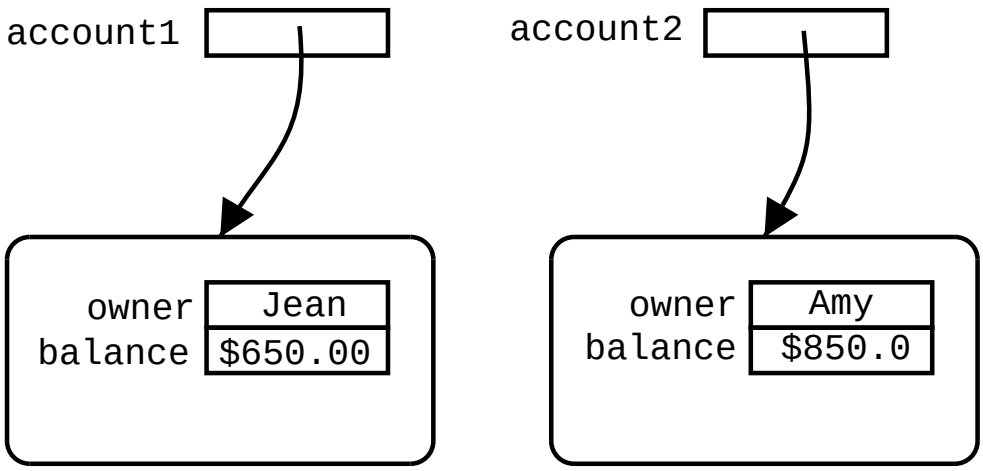
- A bank account has:
  - owner
  - balance
- Given a bank account we can:
  - deposit
  - withdraw

Objects and Classes




```
account1.withdraw(150.00);
```

Objects and Classes



Objects and Classes

```
public class BankAccount {
    String owner;
    double balance;

    void withdraw(double amount)
    {
      // ...
    }

    void deposit(double amount)
    {
      // ...
    }
}
```

Strings, Classes and Objects

- A variable contains either
  - a primitive value (e.g. 2, 3.14, true, ...)
  - or a reference to an object
- The String type is actually a class
- Declaring a variable of a non-primitive type does not create an object of that type:

```
String title;
```

- This declaration only results in allocating (reserving) a memory cell which may hold a reference to a String object

Strings, Classes and Objects

- To create objects we use the `new` operator

```
title = new String(“Trainspotting”);
```

- which is equivalent to

```
title = “Trainspotting”;
```

- But only for Strings

Strings, Classes and Objects

- The operations that can be performed on an object are given by the methods in the class of the object
- The String class has many methods

```
String title;
title = new String("Trainspotting");
title.toLowerCase();
```

- The statement

```
title.toLowerCase();
```

is a *method call* or *method invocation*

Strings, Classes and Objects

- A method call applies an operation on an object
- A method call always has the form

  object`.`*methodname*`(`*parameters*`)`

where `methodname` is a method defined in the class of the object

- We can think of a method call as sending a message to an object

$$\underbrace{\texttt{System.out.}}_{\text{object}}\quad\underbrace{\texttt{println}}_{\text{method}}\quad\underbrace{(\texttt{"text"});}_{\text{parameters}}$$

Strings, Classes and Objects

- Some methods of the String class
  - `charAt`: returns the character of the string at a given position
  - `length`: returns the length of the string
  - `toLowerCase`: returns a copy of the string in lower case
  - `toUpperCase`: returns a copy of the string in uper case
  - `equals`: returns whether the string is equal to another given string
  - `substring`: returns a part of the string given by the parameters
  - etc.

Strings, Classes and Objects

```
public class String {
  //...
  char charAt(int index) { //... }

  int length() { // ... }

  String toLowerCase() { //... }

  String toUpperCase() { //... }

  boolean equals(String str) { // ... }

  String substring(int offset, int endIndex) { //.

  //...
}
```

Strings, Classes and Objects

“	b	o	n	j	o	u	r	”
	0	1	2	3	4	5	6	

Strings, Classes and Objects

```
String name, first_name, last_name;
char initial1, initial2;
int len;

name = new String(“Charles Darwin”);
initial1 = name.charAt(0);
initial2 = name.charAt(8);
len = name.length();
first_name = name.substring(0, 7);
last_name = name.substring(8, len);

System.out.println(first_name);
System.out.println(last_name);
System.out.println(“”+initial1+initial2);
```

Strings, Classes and Objects

```
String name, first_name, last_name;
char initial1, initial2;
int len;

name = new String(“Charles Darwin”);
initial1 = name.charAt(0);
initial2 = name.charAt(8);
len = name.length();
first_name = name.substring(0, 7);
last_name = name.substring(8, len);

System.out.println(first_name);
System.out.println(last_name);
System.out.println(“”+initial1+initial2);
```

Strings, Classes and Objects

```
String name, first_name, last_name;
char initial1, initial2;
int len;

name = new String(“Charles Darwin”);
initial1 = name.charAt(0);
initial2 = name.charAt(8);
len = name.length();
first_name = name.substring(0, 7);
last_name = name.substring(8, len);

System.out.println(first_name);
System.out.println(last_name);
System.out.println(“”+initial1+initial2);
```

Strings, Classes and Objects

```
String name, first_name, last_name;
char initial1, initial2;
int len;

name = new String(“Charles Darwin”);
initial1 = name.charAt(0);
initial2 = name.charAt(8);
len = name.length();
first_name = name.substring(0, 7);
last_name = name.substring(8, len);

System.out.println(first_name);
System.out.println(last_name);
System.out.println(“”+initial1+initial2);
```

Strings, Classes and Objects

```
String name, first_name, last_name;
char initial1, initial2;
int len;

name = new String(“Charles Darwin”);
initial1 = name.charAt(0);
initial2 = name.charAt(8);
len = name.length();
first_name = name.substring(0, 7);
last_name = name.substring(8, len);

System.out.println(first_name);
System.out.println(last_name);
System.out.println(“”+initial1+initial2);
```

Strings, Classes and Objects

```
String name, first_name, last_name;
char initial1, initial2;
int len;

name = new String(“Charles Darwin”);
initial1 = name.charAt(0);
initial2 = name.charAt(8);
len = name.length();
first_name = name.substring(0, 7);
last_name = name.substring(8, len);

System.out.println(first_name);
System.out.println(last_name);
System.out.println(“”+initial1+initial2);
```

Strings, Classes and Objects

```
String name, first_name, last_name;
char initial1, initial2;
int len;

name = new String(“Charles Darwin”);
initial1 = name.charAt(0);
initial2 = name.charAt(8);
len = name.length();
first_name = name.substring(0, 7);
last_name = name.substring(8, len);

System.out.println(first_name);
System.out.println(last_name);
System.out.println(“”+initial1+initial2);
```

Strings, Classes and Objects

```
Charles
Darwin
CD
```

Strings, Classes and Objects

```
String name, first_name, last_name;
char initial1, initial2;
int len;

name = new String(“Charles Darwin”);
initial1 = name.charAt(1);
initial2 = name.charAt(9);
len = name.length();
first_name = name.substring(1, 7);
last_name = name.substring(8, len);

System.out.println(first_name);
System.out.println(last_name);
System.out.println(“”+initial1+initial2);
```

Strings, Classes and Objects

```
harles
Darwin
ha
```

Strings, Classes and Objects

```
String name, first_name, last_name;
char initial1, initial2;
int len;

name = new String(“Charles Darwin”);
initial1 = name.charAt(0);
initial2 = name.charAt(8);
len = name.length();
first_name = name.substring(0, 7);
last_name = name.substring(8, len);

System.out.println(first_name);
System.out.println(last_name);
System.out.println(“”+initial1+initial2);
```

Strings, Classes and Objects

```
String name, first_name, last_name;
char initial1, initial2;
int len;

name = new String(“Charles Darwin”);
len = name.length();
first_name = name.substring(0, 7);
last_name = name.substring(8, len);
initial1 = first_name.charAt(0);
initial2 = last_name.charAt(0);

System.out.println(first_name);
System.out.println(last_name);
System.out.println(“”+initial1+initial2);
```

Strings, Classes and Objects

```
Charles
Darwin
CD
```

Strings, Classes and Objects

- In strings,
  - the first character has index 0
  - the second character has index 1
  - the third character has index 2
  - ...
  - the last character has index l-1, where l is the length of the string

- Example:

```
char c;
String s = "something";
int i = 0;
c = s.charAt(i);
```

- If the parameter of charAt is greater or equal to the length, then a run-time error occurs

Strings, Classes and Objects

- Some method calls can appear as expressions and others as statements

```
String s = "abc";
int n = s.length();
```

- Here, the call to method length is an expression because it occurs in the right-hand side of an assignment

```
System.out.println("abc");
```

- Here, the call to the method println is a statement because it is not being assigned to anything

Class name and files

- A class definition like:

```
public class MyProgram
{
  public static void main(String[] args)
  {
    //...
  }
}
```

Must be in a file named

```
MyProgram.java
```