Iterator

- see the previous slides
- actually, it is generic type

```
interface Iterable<T> {
  Iterator<T> iterator();
}

interface Iterator<T> {
  boolean hasNext();
  T next();
  void remove();
}
```

Reading from a textual file

- package java.io
  - almost everything throws IOException
- sequential reading
- file opening and reading by characters

```
FileReader fr = new FileReader(fileName);
int ch;
while ((ch = fr.read()) != -1) {
  // do something with the character
}
fr.close()
```

- by lines

```
BufferedReader br = new BufferedReader(new
                                  FileReader(fileName));
String line;
while ((line = br.readLine()) != null) {
  ......
}
br.close()
```

Standard input

- System.in
- copying from std input to std output

```
try (BufferedReader input = new BufferedReader(
        new InputStreamReader(System.in))) {
  int c;
  while ((c = input.read()) != -1) {
    System.out.print((char) c);
  }
} catch (IOException ex) {
  System.out.println("IOException occured")
}
```

Assignment 1

- create a class MyString that will serve as a modifiable string
  - it will have at least following methods and constructors
    - MyString()
    - MyString(String str)
    - void append(String str)
    - void append(char ch)
    - void insert(int pos, String str)
    - void insert(int pos, char ch)
    - void delete(int pos, int length)
    - correctly overridden method String toString()
  - it will have an iteraror for iterating over individual characters

Assignment 2

- create a program that prints a textual file split by words, i.e. each word on the new line
  - word separators are white characters (space, new line, tab,...)
  - the name of a file will be given as a program argument
- create a program that prints a textual file justified to a maximal given width (maximal number of characters on the line)
  - the name of a file and number of characters will be given as a program arguments

Tests...

Test 1

Exam test

- What will happen?

```
public class Null {

  public static void main(String[] argv) {
    ((Null) null).hello();
  }

  public static void hello() {
    System.out.println("Hello world!");
  }
}
```

- A program cannot be compiled
- B throws the NullPointerException
- C prints out Hello World!

Test 2

- What is printed out?

```
public class Test01 {

  public static void main(String[] argv) {
    run();
    System.out.println("End");
  }

  public static void run() {
    try {
      run();
    } finally {
      run();
    }
  }
}
```

A cannot be compiled
B End
C nothing
D throws an exception

Test 3

- What is printed out?

```
class A {
  public String className = "A";
}

class B extends A {
  private String className = "B";
}

public class Test02 {
  public static void main(String[] argv) {
    System.out.println(new B().className);
  }
}
```

A cannot be compiled – error in the class B
B cannot be compiled – error in the class Test02
C A
D B
E throws an exception
F prints out something else