

CS 211In

1

Overview

- Doing I/O in Java is a little cumbersome.
- CS211In class:
 - Methods for doing file I/O in a somewhat more convenient if less general way.
 - Makes working with heterogenous data easy
 - Files can have combinations of
 - integers: such as 34 -456
 - words: such as if ad34 er\$rt ert:
 - operators: such as () { }
 - We will call such a thing a token.
 - No support for floating-point numbers or strings
 - Some methods specialized for writing parsers.

2

Word

- Same rules as an identifier in Java except that : is allowed as part of the word.
- Examples:
 - Hello
 - hello
 - u789
 - sdf:

3

Example of file that can be read

- File contains : (34 + -34 wed: -)
- Tokens:
 - Operator: (
 - Integer: 34
 - Operator: +
 - Integer: -34
 - Word: wed:
 - Operator: -
 -
- Note: white space characters are eaten up

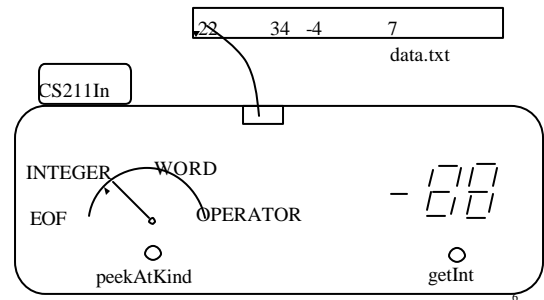
4

Key CS211In Methods

- `peekAtKind()`:
 - look at next token in input without consuming it and return an integer that encodes whether that thing is OPERATOR, WORD, INTEGER, or EOF
- `getInt()`:
 - read an integer from file and return it
 - complain if it is not an integer
- `getWord()` and `getOp()` are similar.

5

Example: add integers in a file



```
public static void main(String[] args)
{CS211In f = new CS211In("data.txt");//create CS211In object
int sum = 0;
while (f.peekAtKind() != CS211In.EOF)
    sum = sum + f.getInt();
System.out.println(sum);
f.close();
}
```

This code assumes there is nothing in file other than integers.

7

With error-checking

```
public static void main(String[] args)
{CS211In f = new CS211In("data.txt");//create CS211In object
int sum = 0;
while (f.peekAtKind() != CS211In.EOF)
    if (f.peekAtKind() == CS211In.INTEGER)
        sum = sum + f.getInt();
    else
        {System.out.println("File contains non-integer data.");
        break;
        }
System.out.println(sum);
f.close();
}
```

8

Example with heterogenous data

- The code shown on next slide reads in heterogenous data from a file one token at a time and prints both the token and its kind on the screen.

File contains : 34 + -34 wed: -

Output:

- Operator: (
- Integer: 34
- Operator: +
- Integer: -34
- Word: wed:
- Operator: -
-

9

```
public static void main(String[] args)
{CS211In f = new CS211In("test1.txt");//create CS211In object
 inputLoop:
 while (true)
 {switch (f.peekAtKind()) {
 case CS211In.EOF: break inputLoop;
 case CS211In.INTEGER: {System.out.println("Integer: " + f.getInt());
 break;
 }
 case CS211In.WORD: {System.out.println("Word: " + f.getWord());
 break;
 }
 case CS211In.OPERATOR: {System.out.println("Operator: " + f.getOp());
 break;
 }
 default: {System.out.println("Unknown kind in file");
 break inputLoop;
 }
 }
 }
 f.close();
 }
```

10

Additional methods in CS211In

- void match(char c):
 - if next token in input is operator c, advance file-pointer past it
 - if next token in input is not c, print error message
- void match(String s):
 - similar to previous method except that it checks for word s

11

- boolean check(char c):
 - if the next token in the input is operator c advance past it and return true.
 - otherwise, do not advance file -pointer and return false.
- boolean check(String s):
 - similar to previous method, except that it checks for word s

12

```

interface CS211InInterface {
    int INTEGER = -1, //returned by peekAtKind at integer token
    WORD = -2, //returned by peekAtKind at word token
    OPERATOR = -3, //returned by peekAtKind at operator token
    EOF = -4, //returned by peekAtKind at end-of-file
    int peekAtKind(); //returns one of the integers above
    int getInt(); //read an integer from file
    String getWord(); //read a word
    char getOp(); //read an operator
    void match(char c); //verify that next thing in file is c
    void match(String s); //verify that next thing in file is s
    boolean check(char c); //is the next thing in file c?
    boolean check(String s); //is the next thing in file s?
    void pushBack(); //back up by one token in the file
    int lineNo(); //where are we?
    void close();
}

```

13