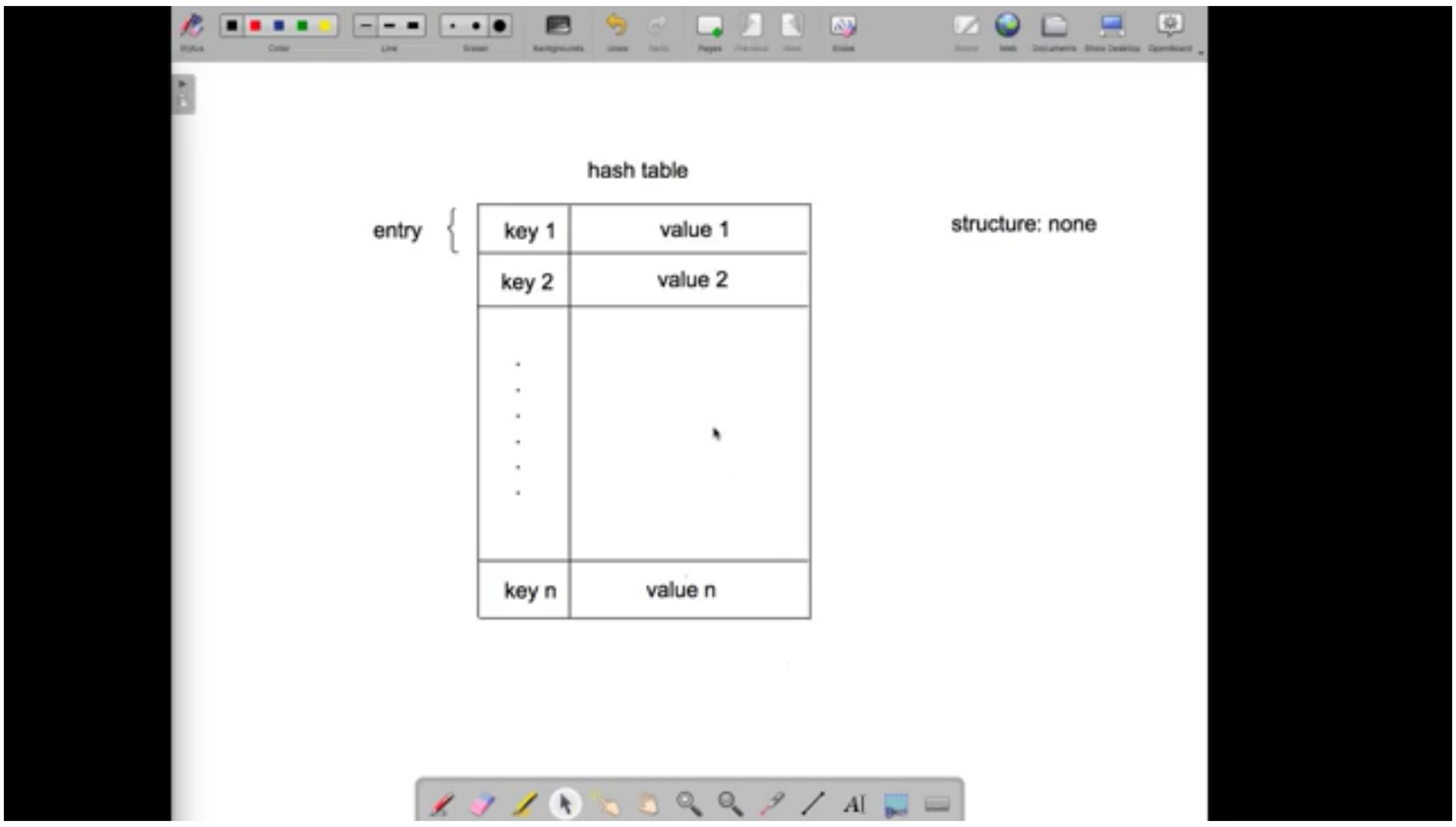
Implementation parser assignment 2

CLASS MODE	LECTURE
Notes	☐
Property	
SLIDES	
TOPIC	

▼ 28: Hash tables, HashMap<K, v>

- Set of entries without structure
- Every entry has
    1. Key
    2. Value
- n keys, n entries → Can all be any object type



```
// Operations:
put(key, value)
boolean contains(key) // Is there an entry with this key?
get(key) --> value // Get value that is stored under this key: either possible or unpossible (null)
```

- In assignment 2: HashMap<K, V>
- Class that defines type K needs to override equals() and hashCode()
- Not needed for assignment, but implementation HashTable:

```
hashfunction f:
hash code = f(key)
```

- HashMap contains default constructor, containsKey(Object key), get(Object key), put(K key, V value)

https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html

▼ 29: EBNF: Extended Backus-Nauer form (what is correct input)

```
stop_symbol = "!" // LHS = stop_symbol = an identifier
```

- Text that is defined (LHS of "=") is a non-terminal
- Text between "" or <> is a terminal. <> are used for description when writing abbreviation of terminal bc it is impossible (<eoln>) or too much work (<all character except the &>)

```
a b // first a, then b
a | b // a or b
[a] // 0 times or 1 time a, thus a is optional
{a} // 0 or more times a
a {a} // 1 or more times a
```

- Example

```
int = [sign] digits // 1. yes/no sign + 2. digit (sign = + or -)
sign = "+" | "-"
digits = digit {digit}
digit = "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"|

e.g. 7, +7, -7, 7777777777
```

▼ 30: Parsing input (Difficult!)

- Design rules for translating EBNF to Java
  1. Every terminal/non-terminal becomes a method
     Call the terminal/non-terminal m and the corresponding method m()
  2. The method m() will read none of the characters that come before or after the characters of m
  3. The method m() will read all of the characters of m and, if necessary, process these characters and return a value
  4. Only exception: if the input is incorrect, an APException will be thrown
     → If m() does not throw an APException, all the characters of m that have been read, were correct and have been processed

```
row = "<" students ">" <eoln>
students = student {"-" student}
student = studentnumber ";" data ";"
studentnumber = digit digit digit
digit = "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"|
data = symbol {symbol}
symbol = <any character except a semicolon> // bc semicolon marks eol so we just keep it simple

example
<001;data student 001;-002;data student 002;>
```

```
class Student
  Student()
  void putStudentnumber(Stringbuffer sb)
  void putData(Stringbuffer sb)

class StudentRow
  StudentRow()
  void add(Student x)

// in the program

boolean nextCharIs (Scanner input, char c)
boolean nextCharIsDigit (Scanner input)
char nextChar (Scanner input)

String row = in.nextLine(); // does not check eoln character
Scanner rowScanner = new Scanner(row);

try {
  StudentRow studentsRow = row(rowScanner);
} catch (APException e) {
  throw new Error("...");
}
```

ImplementationL check if rows are correct

```
row = "<" students ">" <eoln>

studentRow (Scanner input) throws APException {
  character(input,"<"); // if no APException thrown, go to next line
  StudentRow row = students(input);
  character(input,">");
  eoln(input);

  return row;
}

// char method
void character (Scanner input, char c) throws APException {
  if (! nextCharIs(input,c)) {
    throw new APException("...");
  }

  nextChar(input);
}

//eoln method
void eoln (Scanner input) throws APException {
  if (input.hasNext()) {
    throw new APException("...");
  }
}

//studentsmethod
StudentRow students (Scanner input) throws APException {
StudentRow result = new StudentRow();

result.add(student(input));

while (nextCharIs(input, "-")) {
  character(input, "-");
  result.add(student(input));
  }

  return result;
}

//studentmethod

student = studentnumber ";" data ";"

Student student (Scanner input) throws APException {
  Student result = new Student();

  result.putStudentnumber(studentnumber(input));
  character(input, ";");
  result.putData(data(input));
  character(input, ";");

  return result;
}

//studentnumbermethod

studentnumber = digit digit digit
```

```
StringBuffer studentNumber (Scanner input) throws APException {
  Stringbuffer result = new Stringbuffer();

  result.append(digit(input);
  result.append(digit(input);
  result.append(digit(input);

  return result;
}

//digitmethod

digit = "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"|

char digit (Scanner input) throws APException {
  if (! nextCharIsDigit(input)) {
    throw new APException("...");
  }

  return nextChar(input);
}

//datamethod

data = symbol {symbol}

StringBuffer data (Scanner input) throws APException {
  Stringbuffer result = new Stringbuffer();

  result.append(symbol(input));

  while (!nextCharIs(input,";")) {
    result.append(symbol(input)); // stop reading symbols when we read ";"
  }

  return result;
}

//symbolmethod = method that reads 1 symbol

symbol = <any character except a semicolon>

char symbol (Scanner input) throws APException {
  if (nextCharIs(input, ";")) {
    throw new APException("...");
  }

  return nextChar(input);
}
```