CSCI3170 Introduction to Database Systems

Tutorial 2 – Introduction to Java (II)

Outline

- More on Java Basic Syntax
- Introduction to Java IO
- Exception Handling
- Example: A Simple File Viewer
- More Operations on Java
- Introduction to JDBC

Local Variables

- In a Java method, local variables can be created for storing data temporary.
- After a method finish its execution, all local variables will be cleared.
- Example:

```
public static void main (String args[]) {
   String thisLine;
}
```

Constructor

- A constructor is a method which can initialize an object just right after it is created.
- Example:

```
class Student{
  private String Name;
  private String Major;

  public Student (String Name, String Major){
    this.Name = Name;
    this.Major = Major;
  }
}
```

Use “this” to specify the field instead of local variable

INTRODUCTION TO JAVA IO

Java Standard IO

- Standard Output (System.out)

```
System.out.print()
```

  - Print a string to the console through buffer

```
System.out.println()
```

  - Print a string to the console with a newline character through buffer

Java Standard IO (2)

- Standard Error (System.err)

```
System.err.print()
```

  - Print a string to the console immediately

```
System.err.println()
```

  - Print a string to the console with a newline character immediately

Note: You should use System.err for printing debug messages since printing message via System.out may not work when a program has an run time error.

Java Standard Output (2)

- Example

```
System.out.print("Hello World");
```

```
System.out.println("1 + 1 = " + 2);
```

```
System.err.print(2);
```

```
System.err.println('c');
```

Java Standard Input

- Standard Input (System.in)

```
import java.io.*;
```

  - Import all Java standard library on IO

```
BufferedReader in = new BufferedReader
(new InputStreamReader(System.in));
```

  - Create a Object for Reading

Java File Input

- File Input (FileReader)

```
import java.io.*;
```

  – Import all Java standard library on IO

```
BufferedReader in = new BufferedReader
(new FileReader(new File("filename")));
```

  – Create a Object for Reading

Java Input Reader

- To read string

```
String str = in.readLine();
```

- To read a charatcer

```
char c = in.read();
```

- To read numbers
  - read String then convert to appropriate type using

```
Integer.parseInt()
```

```
Double.parseDouble()
```

EXCEPTION HANDLING

Exception Handling

- An appropriate “exception handler” takes over when a run-time error occurs

```
try {
  <Statement(s)>
} catch (<exception type> <name>) {
  <Error Handling Statement(s)>
} finally {
  /* this will be executed after normal execution
      or execution of an exception handler */
  <Statement(s)>
}
```

EXAMPLE: A SIMPLE FILE VIEWER

Example: A Simple File Viewer

```
import java.io.*;
class Example{
  public static void main (String args[]) {
    String thisLine;
    try {
      BufferedReader br = null;
      br = new BufferedReader(new FileReader("1.txt"));
      while ((thisLine = br.readLine()) != null)
        System.out.println(thisLine);
    }catch (IOException e){
       System.err.println("Error: " + e);
    }
  }
}
```

Create a Buffered Reader Object for accessing files

Example: A Simple File Viewer (1)

```
import java.io.*;
class example{
  public static void main (String args[]) {
    String thisLine;
    try {
      BufferedReader br = null;
      br = new BufferedReader(new FileReader("1.txt"));
      while ((thisLine = br.readLine()) != null)
        System.out.println(thisLine);
    }catch (IOException e){
       System.err.println("Error: " + e);
    }
  }
}
```

Iteratively Read a Line from the file 1.txt

Example: A Simple File Viewer (2)

```
import java.io.*;
class Example{
  public static void main (String args[]) {
    String thisLine;
    try {
      BufferedReader br = null;
      br = new BufferedReader(new FileReader("1.txt"));
      while ((thisLine = br.readLine()) != null)
        System.out.println(thisLine);
    }catch (IOException e){
        System.err.println("Error: " +
    }
  }
}
```

Print text to the terminal via standard output

Example: A Simple File Viewer (3)

```
import java.io.*;
class Example{
  public static void main (String args[]) {
    String thisLine;
    try {
      BufferedReader br = null;
      br = new BufferedReader(new FileReader("1.txt"));
      while ((thisLine = br.readLine()) != null)
        System.out.println(thisLine);
    }catch (IOException e){
       System.err.println("Error: " + e);
    }
  }
}
```

Initialize a exception handler for the File IO

Example: A Simple File Viewer (4)

```
import java.io.*;
class Example{
  public static void main (String args[]) {
    String thisLine;
    try {
      BufferedReader br = null;
      br = new BufferedReader(new FileReader("1.txt"));
      while ((thisLine = br.readLine()) != null)
        System.out.println(thisLine);
    }catch (IOException e){
      System.err.println("Error: " + e);
    }
  }
}
```

Handle the error by printing text to the terminal via standard error

MORE OPERATIONS ON JAVA

More Operations on Java

- String Comparison

```
String str = "CSCI3170";
if(str.equals("CSCI3170")) {
   …
}
```

- Split a string into an array based on a delimiter

```
String str = "boo:and";
String[] result= str.split(":");
System.out.println("(1)" + result[0]);
System.out.println("(2)" + result[1]);
```

More Operations on Java (2)

- Getting the date of a local PC as a string

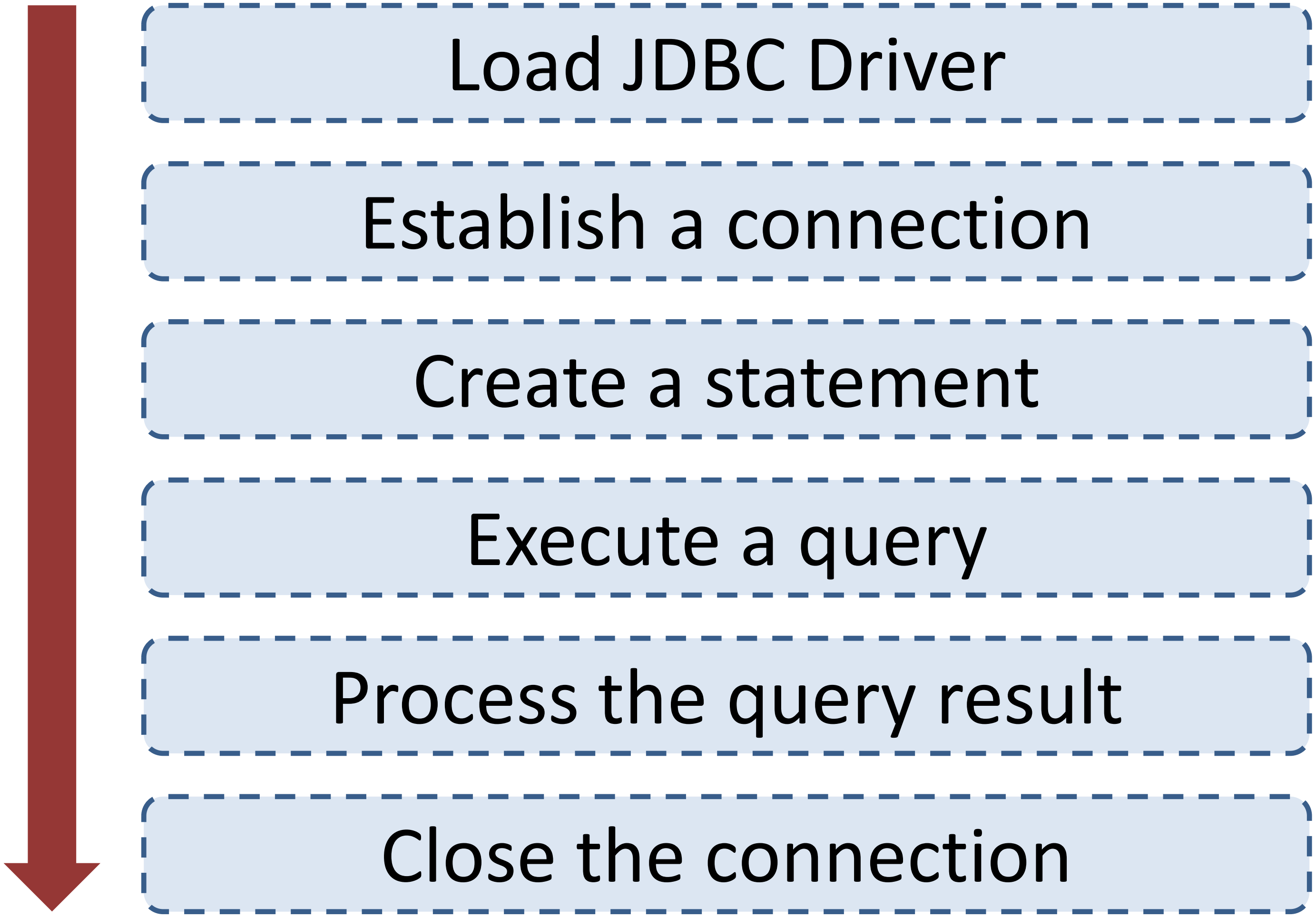
```
Calendar cal = Calendar.getInstance();
SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
String dateInStr = sdf.format(cal.getTime());
```

INTRODUCTION TO JDBC

JDBC

- Stand for Java Database Connectivity
- A Java API for accessing different kind of data
- Manage three activities
  - Connect to a database
  - Send queries to the database
  - Retrieve results from the database

STEPS



Load JDBC Driver

Establish a connection

Create a statement

Execute a query

Process the query result

Close the connection

Load JDBC Driver

- Use **mySQL_JDBC.jar** that we provide
  - Add -classpath for running the program

```
Java -classpath ./mySQL_JDBC.jar:./ <class_file>
```

- Importing required packages for JDBC API

```
import java.sql.*;
```

Establish a Connection

- Load the JDBC Driver for Oracle DBMS

```
try {
   Class.forName("com.mysql.jdbc.Driver");
} catch(Exception x) {
   System.err.println("Unable to load the driver class!");
}
```

- Establish a Connection

```
Connection conn = DriverManager.getConnection(
"jdbc:mysql://projgw.cse.cuhk.edu.hk:2712/username?autoRe
connect=true&useSSL=false", "username", "password");
```

Create and execute a statement

- Create a statement object

```
Statement stmt = conn.createStatement();
```

- Execute a statement using the object

```
stmt.executeUpdate("CREATE TABLE Student " +
                   "(UserID VARCHAR(10), " +
                   "Password VARCHAR(8))");
```

```
ResultSet rs = stmt.executeQuery("SELECT * FROM temp");
```

Process the query result

- Retrieve data from result set

```
String user_id;
String password;

while (rs.next()){
      user_id = rs.getString(1);
      password = rs.getString(2);
}
```

rs.next() moves the cursor down one row from its current position

JDBC Datatypes

NUMBER

JDBC Type	Java Type	Method
INT	int	getInt
REAL	float	getFloat
FLOAT	double	getDouble
DOUBLE	double	getDouble
CHAR	String	getString
VARCHAR	String	getString
DATE	java.sql.Date	getDate
TIMESTAMP	TIMESTAMP	getTimeStamp

The Use of PreparedStatement

- For handling a large number of records

```
PreparedStatement pstmt = con.prepareStatement(
"INSERT INTO Student VALUES (?, ?)");

for (int i = 0, i < student.length, i++){
    pstmt.setString(1, student[i][0]);
    pstmt.setString(2, student[i][1]);
    pstmt.executeUpdate();
}
```

□ Give a better performance as the SQL statement only needs to be compiled once

□ Is more secure as it can prevent some (but not all) SQL injections.

Close the connection

- Avoid holding unnecessary resourses

```
/* destroy the result set object  */
rs.close();

/* destroy the statement object */
stmt.close() ;

/* destroy the prepared statement object */
pstmt.close();

/* destroy the connection */
conn.close();
```