Universitetet i Agder
Fakultet for økonomi- og samfunnsfag

EKSAMEN

Emnekode:	**IS-202**
Emnenavn:	**Topics in programmering**
Dato:	**15. December** 2008
Varighet:	**0900-1300**
Antall sider inkl. forside:	4
Målform:	**English**
Tillatte hjelpemidler:	**All printed matter and** handwritten notes.
Merknader:	The problems count equally towards the grade.

Problem 1: The Customer Registration page

In this problem set we will look at some aspects of customer registration for a web shop. To register as a customer the web shop user must provide information such as name and address. This is done by filling in an html form in web page.
Write the html code for the registration form. You need input fields for name, address, username and password.

Problem 2: The Customer Registration servlet

When the user has filled in the form and clicks the submit button, it will be sent to the server and handled by a customer registration servlet.
Write the customer registration servlet. The servlet must do the following:

1. Check that all fields in the form have been filled in
2. Create a new Customer object (see appendix 1)
3. Save the customer object to the database by calling the save() method.
4. Write a confirmation page that will be returned to the browser.

A reminder of useful classes and methods can be found in appendix 3.

Problem 3: Saving the Customer to the database

Finish the save() method in the Customer class. The method should store the attributes of the Customer object in the corresponding columns in the CUSTOMER table (see appendix 2).
A reminder of useful classes and methods can be found in appendix 3.

Problem 4:

Explain what the program "ant" does, and how and why we use it?

Appendix 1: Class Customer

```
/**
 * Class Customer. Each object of this class represents a web shop
 * customer.
 */
public class Customer {
    private String username;
    private String password;
    private String fullname;
    private String address;


    /**
     * Constructor
     * @param username The username used to log in to the shop
     * @param password The password
     * @param fullname The customer's real name
     * @param address The customer's delivery/billing address
     */
    public Customer(String username, String password,
                    String fullname, String address) {
        this.username = username;
        this.password = password;
        this.fullname = fullname;
        this.address = address;
    }


    /**
     * Save the customer data to the database.
     * @param con The JDBC connection to use
     */
    public void save(Connection con) {
        // problem 3
    }


    // A customer class for a real webshop will have more methods,
    // but they are not needed for this problem set.
}
```

Appendix 2: Customer Table

```
CREATE TABLE customer (
    username VARCHAR(128) NOT NULL,
    password VARCHAR(32) NOT NULL,
    fullname VARCHAR(128) NOT NULL,
    address VARCHAR(128) NOT NULL,

    CONSTRAINT customer_pk PRIMARY KEY(username)
);
```

This sql code can be used in a PreparedStatement to insert a new row:
INSERT INTO customer (username, password, fullname, address) VALUES (? ? ? ?)

Appendix 3: Library classes summary

javax.servlet.http.HttpServlet

void doPost(HttpServletRequest req, HttpServletResponse resp) - Called by the server to allow a servlet to handle a POST request.

javax.servlet.http.HttpServletRequest

String getParameter(String name) - Returns the value of a request parameter as a String, or null if the parameter does not exist.

javax.servlet.http.HttpServletResponse

PrintWriter getWriter() - Returns a PrintWriter object that can send character text to the client.

class java.sql.Connection

Statement createStatement() - Creates a Statement object for sending SQL statements to the database.
PreparedStatement prepareStatement(String sql) - Creates a PreparedStatement object for sending parameterized SQL statements to the database.

java.sql.Statement

int executeUpdate(String sql) - Executes the given SQL statement, which may be an INSERT, UPDATE, or DELETE statement or an SQL statement that returns nothing, such as an SQL DDL statement

java.sql.PreparedStatement

int executeUpdate() - Executes the SQL statement in this PreparedStatement object, which must be an SQL Data Manipulation Language (DML) statement, such as INSERT, UPDATE or DELETE; or an SQL statement that returns nothing, such as a DDL statement.
void setString(int parameterIndex, String x) - Sets the designated parameter to the given Java String value.