Advanced Java Exercises

Generics

Exercise 1: Write a generic method to exchange the positions of two different elements in an array.

Exercise 2: Consider the following classes:

```
public class AnimalHouse<E> {
   private E animal;
   public void setAnimal(E x) {
      animal = x;
   }
   public E getAnimal() {
      return animal;
   }
}
public class Animal{
}
public class Cat extends Animal {
}
public class Dog extends Animal {
}
```

For the following code snippets, identify whether the code:

fails to compile,

compiles with a warning,

generates an error at runtime, or

none of the above (compiles and runs without problem.)

Question 1a. AnimalHouse<Animal> house = new AnimalHouse<Cat>();

Question 1b. AnimalHouse<Cat> house = new AnimalHouse<Animal>();

Question 1c. AnimalHouse<?> house = new AnimalHouse<Cat>();

Question 1d. AnimalHouse house = new AnimalHouse();

Collections

Exercise 1: Create Employee class with two members id, name and method displayEmployee. Create Employee objects and add them into ArrayList. Use get method and retrieve the Employee objects from list and display id and name values

Exercise 2: Modify Exercise 1 so you use an Iterator to move through the List while calling displayEmployee( ).

Exercise 3: Take the Employee class in Exercise 1 and put it into a Map instead, associating the name of the Employee as a String (the key) for each Employee (the value) you put in the table. Get an Iterator for the keySet( ) and use it to move through the Map, looking up the Employee for each key and printing out the key and telling the Employee object to displayEmployee( ).

Exercise 3: Create a List (ArrayList) and fill it using Collections2.countries. Sort the list and print it, then apply Collections.shuffle( ) to the list repeatedly, printing it each time so that you can see how the shuffle( ) method randomizes the list differently each time.

Exercise 4: Fill a HashMap with key-value pairs. Print the results to show ordering by hash code. Extract the pairs, sort by key, and place the result into a LinkedHashMap. Show that the insertion order is maintained.

Exercise 5: Write a program to sort Employee objects based on highest salary using Comparator.

Exercise 6: Write a program to eliminate duplicate keys (user defined objects) with Hashtable?

Hint: Use user defined objects (Employee) as keys. To avoid duplicates keys you have to implement equals and hashcode methods. (To successfully store and retrieve objects from a hashtable, the objects used as keys must implement the hashCode method and the equals method.).

Exercise 7: Write a program and get all key-value pair as java.util.Map.Entry objects. java.util.Map.Entry class provides getter methods to access key-value details. The method entrySet() provides all entries as set object. And use enhanced for loop to transverse through all the elements of Set.

JDBC

Exercise 1: Log into the mysql server and create a sample database, table, and then add some rows to the table. Later create a simple program to connect to database and query the table.

Exercise 2: Create Employee table with id, name, department and add rows through java JDBC using statement and prepare statements.

Exercise 3: Use Employee table and perform updating employee

Exercise 4: Find employee with highest salary.

Exercise 5: Use Employee table and perform deleting an employee operation.

Java Beans

Exercise 1: Create a simple class with name Department and member variables deptId and deptName. Create two setter methods and getter methods. Create a Test class, in main method of this class, create Department objects and set values through setter methods and display the values using getter methods of Department class.

XML

Exercise 1: Create a Emplyee.xml with the following content in it

```
<?xml version="1.0" encoding="UTF-8"?>

<Personnel>

  <Employee type="permanent">

      <Name>Seagull</Name>

      <Id>3674</Id>

      <Age>34</Age>

   </Employee>

  <Employee type="contract">

      <Name>Robin</Name>

      <Id>3675</Id>

      <Age>25</Age>

    </Employee>

  <Employee type="permanent">

      <Name>Crow</Name>
```

```
    <Id>3676</Id>
    <Age>28</Age>
  </Employee>
</Personnel>
```

Create java programs to parse the xml content and display employee details, one program using DOM parser and another one using SAX parser.

Exercise 2: Write a program to generate a XML file with the following content.To keep the example simple this program generates XML from a list preloaded with hard coded data. The output will be book.xml file with the following content.

```
<?xml version="1.0" encoding="UTF-8"?>
<Books>
  <Book Subject="Java 1.5">
    <Author>Kathy Sierra .. etc</Author>
    <Title>Head First Java</Title>
  </Book>
  <Book Subject="Java Architect">
    <Author>Kathy Sierra .. etc</Author>
    <Title>Head First Design Patterns</Title>
  </Book>
</Books>
```

JSP

Exercise 1: Write a page “MiniAdd.jsp” that carries out simple additions of two numbers. The JSP

should:

present a form where the user can specify two numbers,

contain a submit button that posts back to the same page (i.e. “miniadd.jsp”),

show the result of the requested addition

Exercise 2: Create a simple HTML with form and two input fields like First Name and Last Name. Submit the data to a jsp where the form data is retrieved and displayed. Create form with GET method and another with POST method.

Exercise 3: Create an Employee bean with id and name member variables. Access Employee bean objects in a sample jsp and display values.

Exercise 4: Create a JSP and set some request attributes and access them in another jsp file and display

Exercise 5: Create a JSP and access session object and store some attributes in it.

Exercise 6: Create a header.jsp and add one print statement in it. Create main.jsp and include header.jsp in it.

Exercise 7: Create a simple custom tag that displays hello message and use this tag in two jsps and display hello message.

Exercise 8: Create jsp that performs redirection to www.google.com

Exercise 9: Create a jsp that explicitly throws an error and create an error.jsp to display the error.

Servlets

Exercise 1: Create a simple Servlet to display a message using PrintWriter object.

Exercise 2: Create a simple HTML with form and two input fields like First Name and Last Name. Submit the data to a servlet where the form data is retrieved and displayed. Create form with GET method and another with POST method.

Exercise 3: Wrtie a servlet that uses getHeaderNames() method of HttpServletRequest to read the HTTP header information and display

Exercise 4: Write a servlet to display current time and refresh at every 5 seconds.

Exercise 5: Write a servlet to forward a request to another Servlet.

Exercise 6: Write a servlet to forward a request to another jsp.

Exercise 7: Write a servlet and initialize the serlvet through servlet configuration object

Exercise 8: Write a servelt and add some attributes into Servlet Context object and access them in another servlet

Exercise 9: Update Exercise 2 problem, create cookies with first name and last name and add them to response.

Exercise 10: Create a serlvet that describes how to use the HttpSession object to find out the creation time and the last-accessed time for a session. We would associate a new session with the request if one does not already exist.

Exercise 11: Create a filter to display a message, and create one jsp and servlet. Try to access jsp and servlet and observe how many times the filter message gets displayed.