COMP/EECE 7012/8012

Exam 2

Spring 2013

Name: _____

Rules:

- No potty breaks.
- Turn off cell phones/devices.
- Closed book, closed note, closed neighbor.

- NEW! Do not write on the backs of pages. Ask me for extra sheets of paper if you need them.

Reminders:

- Verify that you have 9 pages of questions and 5 of figures.
- Don't forget to write your name.
- Read each question carefully.
- Don't forget to answer every question.

Additional Items:

- For questions that involve writing code:
  - You may omit `import` statements.
  - You may omit exception-handling code.

1. [5pts] Match each item on the right to the development approach that the item best characterizes.

Big Bang •

Iterative •

- Collects <u>all</u> requirements before building anything
- Has tight feedback loop
- Produces working software quickly
- Spends a long time building the wrong software
- Assumes that requirements are unstable

2. [3pts] What three things does great software deliver?

a. _____

b. _____

c. _____

3. [4pts] Of the two user stories in Figure 1, which was better written? Explain your answer.

4. [2pts] All else being equal, which of the USs in Figure 1 most likely has the more accurate estimate?

5. [4pts] What <u>two</u> things are wrong with the following series of steps?

   (1) First, the developers solicit user stories from the customer.
   (2) Next, the developers assign a priority level to each user story.
   (3) Next, the developers estimate the effort required to implement each user story.

6. [2pts] Which of the following techniques is used for estimating effort?

   a. Role playing

   b. Blueskying

   c. Planning poker

   d. Observation

   e. None of the above

7. [4pts] If your team planned to do 45 days worth of work, but it actually took them 50 days, what is your team's velocity?

8. [6pts] After your team chooses the USs to implement in an iteration, but before the team begins implementing, what three things must the team do?

9. [2pts] Based on the burn-down graph in Figure 2, did the team finish the iteration ahead of schedule, behind schedule, or on schedule?

10. [2pts] If you want to make radical changes to your team's project and don't want to impact the rest of the team, you should implement your changes in …

    a. … a tag.

    b. … the trunk.

    c. … the root.

    d. … a branch.

    e. None of the above

Remember the snack-ordering system from Exam 1? It's back! The code has changed, though. Updated screenshots and code can be found in Figure 3–Figure 15. Use these figures to answer the remaining questions.

11. [8pts] Given the Java code in the figures, draw a class diagram that models the code. Include only the following classes in your diagram: **SnackOrder**, **SnackOrderDao**, **SnackItem**, **DrinkItem**, and **ChipsItem**. You need not include getter/setter methods that only assign/return instance variables.

12. [10pts] Refactor the JSPs (Figure 6–Figure 9) so that they better obey the DRY principle (i.e., write code). Of the three JSPs, you may redefine only one (e.g., **orderError.jsp**) as long as it's clear from that one what changes you would make to the others. Hint: you may also need to create new files. Recall that the JSP directive tag **<%@ include file="foo.jsp" %>** inserts the contents of **foo.jsp**.

13. [10pts] Refactor the **SnackOrder** class (i.e., write code) so that it better obeys the SRP principle. Hint #1: there should be no need to modify your code if new types of snacks are introduced (e.g., candy). Hint #2: the problem is that the class knows too much.

14. [8pts] Define a white-box test suite that covers all possible paths through the **OrderSnackServlet**'s **doPost** method (Figure 10). You may use plain English, but it must be unambiguous what the inputs of each test case would be like.

Title: Animated Buttons

Description: Use jQuery to animate buttons.

Estimate: 2 days

Title: Review Flight

Description: A user will be able to leave a review for a shuttle flight they have been on.

Estimate: 20 days

Figure 1. Two example user stories.

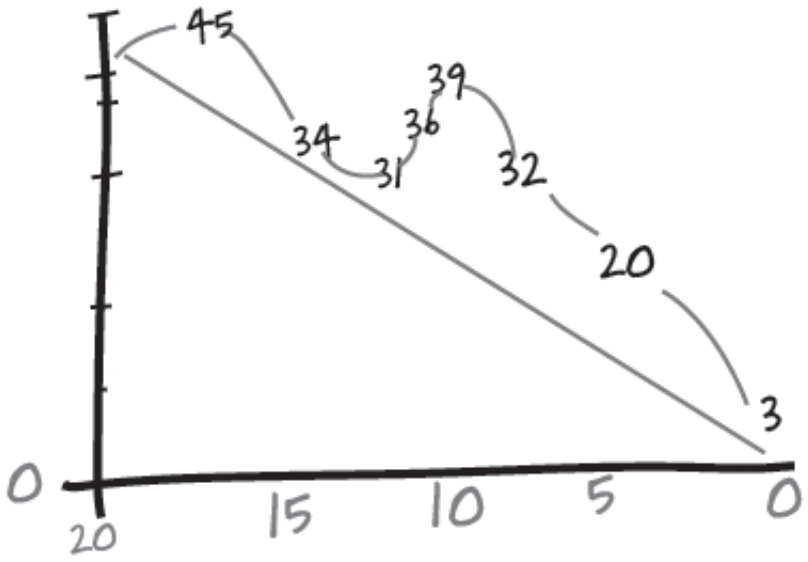

Figure 2. Example burn-down graph.

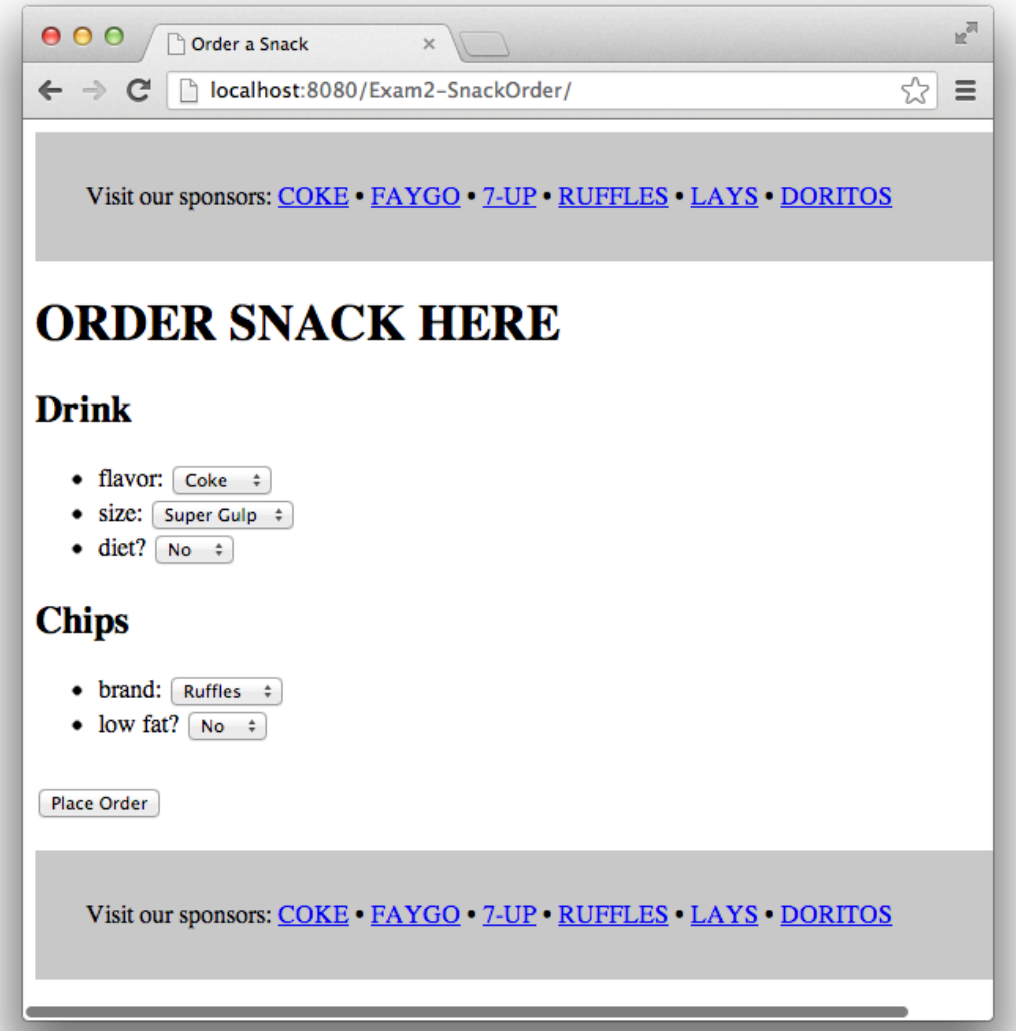

Figure 3. The snack-order system's index.jsp page.

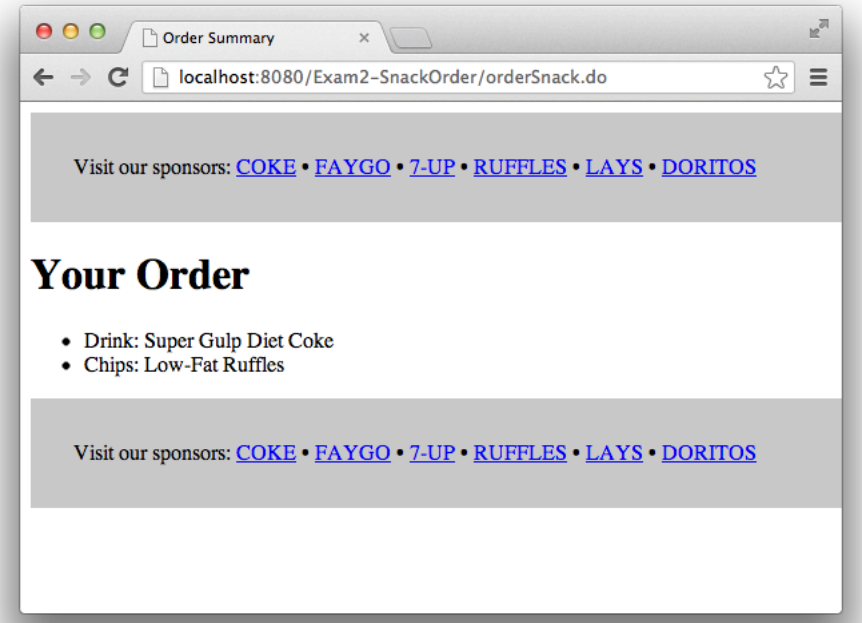

Figure 4. The snack-order system's orderSummary.jsp page.

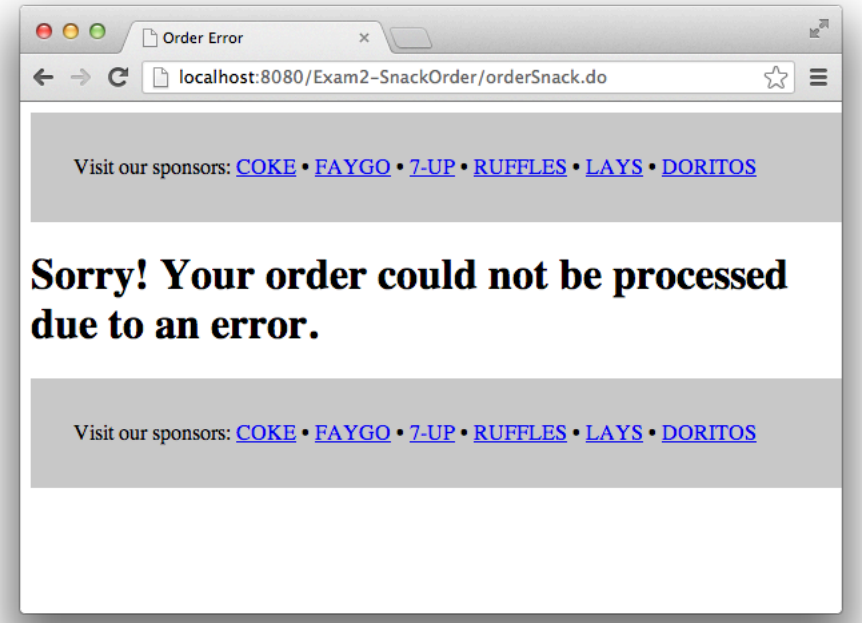

Figure 5. The snack-order system's orderError.jsp page.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Order a Snack</title>
</head>
<body>

<!-- STANDARD ADVERTISEMENT BEGIN -->
<div style="width: 100%; background-color: #CCC; margin: 0; padding: 2em;">
Visit our sponsors:
<a href="http://www.coke.com">COKE</a> &bull;
<a href="http://www.faygo.com">FAYGO</a> &bull;
<a href="http://www.7up.com">7-UP</a> &bull;
<a href="http://www.fritolay.com/our-snacks/ruffles-original.html">RUFFLES</a> &bull;
<a href="http://www.lays.com">LAYS</a> &bull;
<a href="http://www.doritos.com">DORITOS</a>
</div>
<!-- STANDARD ADVERTISEMENT END -->

<form method="post" action="orderSnack.do">
<h1>ORDER SNACK HERE</h1>
<h2>Drink</h2>
<ul>
<li>flavor:
<select name="drinkFlavor">
<option value="Coke" selected>Coke</option>
<option value="Faygo">Faygo</option>
<option value="7-Up">7-Up</option>
</select></li>
<li>size:
<select name="drinkSize">
<option value="Small">Small</option>
<option value="Medium">Medium</option>
<option value="Large">Large</option>
<option value="Super Gulp" selected>Super Gulp</option>
</select></li>
<li>diet?
<select name="drinkDiet">
<option value="Yes">Yes</option>
<option value="No" selected>No</option>
</select></li>
</ul>
```

Figure 6. index.jsp (part 1 of 2)

```
<h2>Chips</h2>
<ul>
<li>brand:
<select name="chipsBrand">
<option value="Ruffles" selected>Ruffles</option>
<option value="Lays">Lays</option>
<option value="Doritos">Doritos</option>
</select></li>
<li>low fat?
<select name="chipsLowFat">
<option value="Yes">Yes</option>
<option value="No" selected>No</option>
</select></li>
</ul>
<h2><input type="submit" value="Place Order"></h2>
</form>

<!-- STANDARD ADVERTISEMENT BEGIN -->
<div style="width: 100%; background-color: #CCC; margin: 0; padding: 2em;">
Visit our sponsors:
<a href="http://www.coke.com">COKE</a> &bull;
<a href="http://www.faygo.com">FAYGO</a> &bull;
<a href="http://www.7up.com">7-UP</a> &bull;
<a href="http://www.fritolay.com/our-snacks/ruffles-original.html">RUFFLES</a> &bull;
<a href="http://www.lays.com">LAYS</a> &bull;
<a href="http://www.doritos.com">DORITOS</a>
</div>
<!-- STANDARD ADVERTISEMENT END -->

</body>
</html>
```

Figure 7. index.jsp (part 2 of 2)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" import="com.snack.model.*" import="java.util.Vector" %>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Order Summary</title>
</head>
<body>

<!-- STANDARD ADVERTISEMENT BEGIN -->
<div style="width: 100%; background-color: #CCC; margin: 0; padding: 2em;">
Visit our sponsors:
<a href="http://www.coke.com">COKE</a> &bull;
<a href="http://www.faygo.com">FAYGO</a> &bull;
<a href="http://www.7up.com">7-UP</a> &bull;
<a href="http://www.fritolay.com/our-snacks/ruffles-original.html">RUFFLES</a> &bull;
<a href="http://www.lays.com">LAYS</a> &bull;
<a href="http://www.doritos.com">DORITOS</a>
</div>
<!-- STANDARD ADVERTISEMENT END -->

<h1>Your Order</h1>

<ul>
<%
SnackOrder order = (SnackOrder) request.getAttribute("order");
Vector<SnackItem> items = order.getItems();
%>
<li>Drink: <%= items.get(0).getName() %>
<li>Chips: <%= items.get(1).getName() %>
</ul>

<!-- STANDARD ADVERTISEMENT BEGIN -->
<div style="width: 100%; background-color: #CCC; margin: 0; padding: 2em;">
Visit our sponsors:
<a href="http://www.coke.com">COKE</a> &bull;
<a href="http://www.faygo.com">FAYGO</a> &bull;
<a href="http://www.7up.com">7-UP</a> &bull;
<a href="http://www.fritolay.com/our-snacks/ruffles-original.html">RUFFLES</a> &bull;
<a href="http://www.lays.com">LAYS</a> &bull;
<a href="http://www.doritos.com">DORITOS</a>
</div>
<!-- STANDARD ADVERTISEMENT END -->

</body>
</html>
```

Figure 8. orderSummary.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Order Error</title>
</head>
<body>

<!-- STANDARD ADVERTISEMENT BEGIN -->
<div style="width: 100%; background-color: #CCC; margin: 0; padding: 2em;">
Visit our sponsors:
<a href="http://www.coke.com">COKE</a> &bull;
<a href="http://www.faygo.com">FAYGO</a> &bull;
<a href="http://www.7up.com">7-UP</a> &bull;
<a href="http://www.fritolay.com/our-snacks/ruffles-original.html">RUFFLES</a> &bull;
<a href="http://www.lays.com">LAYS</a> &bull;
<a href="http://www.doritos.com">DORITOS</a>
</div>
<!-- STANDARD ADVERTISEMENT END -->

<h1>Sorry! Your order could not be processed due to an error.</h1>

<!-- STANDARD ADVERTISEMENT BEGIN -->
<div style="width: 100%; background-color: #CCC; margin: 0; padding: 2em;">
Visit our sponsors:
<a href="http://www.coke.com">COKE</a> &bull;
<a href="http://www.faygo.com">FAYGO</a> &bull;
<a href="http://www.7up.com">7-UP</a> &bull;
<a href="http://www.fritolay.com/our-snacks/ruffles-original.html">RUFFLES</a> &bull;
<a href="http://www.lays.com">LAYS</a> &bull;
<a href="http://www.doritos.com">DORITOS</a>
</div>
<!-- STANDARD ADVERTISEMENT END -->

</body>
</html>
```

Figure 9. orderError.jsp

```
@WebServlet("/orderSnack.do")
public class OrderSnackServlet extends HttpServlet {
	private static final long serialVersionUID = 1L;

	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
		DrinkItem drink = new DrinkItem();
		drink.setFlavor(request.getParameter("drinkFlavor"));
		drink.setSize(request.getParameter("drinkSize"));
		String drinkDietStr = request.getParameter("drinkDiet");
		if (drinkDietStr.equals("Yes")) {
			drink.setDiet(true);
		} else {
			drink.setDiet(false);
		}

		ChipsItem chips = new ChipsItem();
		chips.setBrand(request.getParameter("chipsBrand"));
		String chipsLowFatStr = request.getParameter("chipsLowFat");
		if (chipsLowFatStr.equals("Yes")) {
			chips.setLowFat(true);
		} else {
			chips.setLowFat(false);
		}

		SnackOrder order = new SnackOrder();
		order.addDrinkItem(drink);
		order.addChipsItem(chips);
		request.setAttribute("order", order);

		SnackOrderDao dao = new SnackOrderDao();

		if (dao.insertOrder(order) == -1) {
			request.getRequestDispatcher("WEB-INF/orderError.jsp").forward(request,
response);
		} else {
			request.getRequestDispatcher("WEB-INF/orderSummary.jsp").forward(request,
response);
		}
	}

}
```

Figure 10. OrderSnackServlet.java

```
public class SnackOrder {

	private DrinkItem drink = null;
	private ChipsItem chips = null;

	public void addDrinkItem(DrinkItem drink) { this.drink = drink; }
	public void addChipsItem(ChipsItem chips) { this.chips = chips; }

	public Vector<SnackItem> getItems() {
		Vector<SnackItem> result = new Vector<SnackItem>();
		result.addElement(drink);
		result.addElement(chips);
		return result;
	}

}
```

Figure 11. SnackOrder.java

```
public class SnackOrderDao {

	/**
	 * Returns newly created order number or -1 on error.
	 */
	public int insertOrder(SnackOrder order) {
		...
	}

	...
}
```

Figure 12. SnackOrderDao.java

```
public interface SnackItem {
	public String getName();
}
```

Figure 13. SnackItem.java

```
public class DrinkItem implements SnackItem {

	private String flavor = "";
	private String size = "";
	private boolean diet = false;

	public void setFlavor(String flavor) { this.flavor = flavor; }
	public void setSize(String size) { this.size = size; }
	public void setDiet(boolean diet) { this.diet = diet; }

	public String getFlavor() { return flavor; }
	public String getSize() { return size; }
	public boolean isDiet() { return diet; }

	public String getName() {
		return size + " " + (diet?"Diet ":"") + flavor;
	}

}
```

Figure 14. DrinkItem.java

```
public class ChipsItem implements SnackItem {

	private String brand = "";
	private boolean lowFat = false;

	public void setBrand(String brand) { this.brand = brand; }
	public void setLowFat(boolean lowFat) { this.lowFat = lowFat; }

	public String getBrand() { return brand; }
	public boolean isLowFat() { return lowFat; }

	public String getName() {
		return (lowFat?"Low-Fat ":"") + brand;
	}

}
```

Figure 15. ChipsItem.java