PalmGrocer Electronic Cookbook Elaboration 1 Document

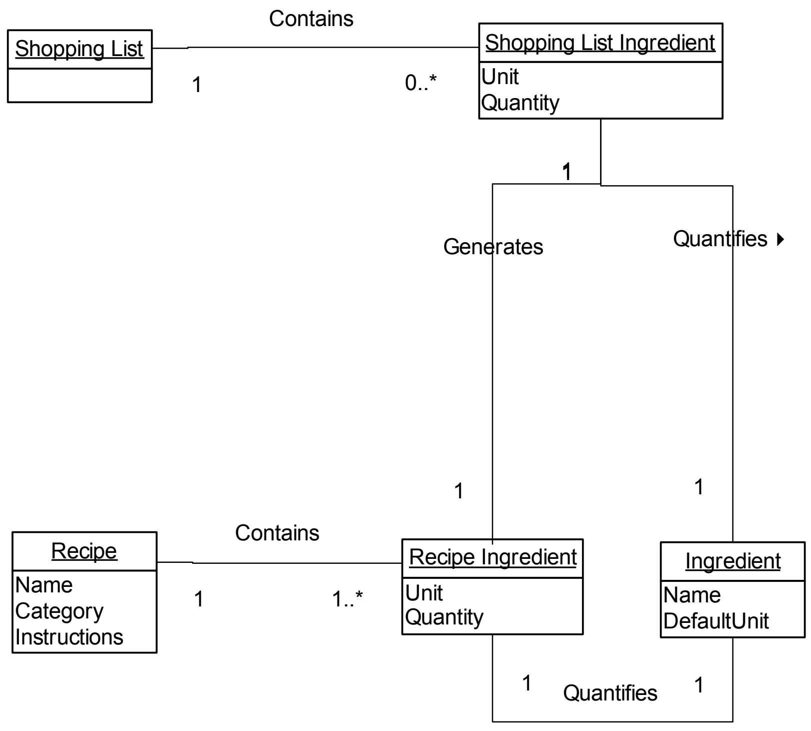
November 10, 2004

By:
Andrew Alford
Andrej Jechropov
Sharmila Pandith
Adam Zimmerman

Table of Contents

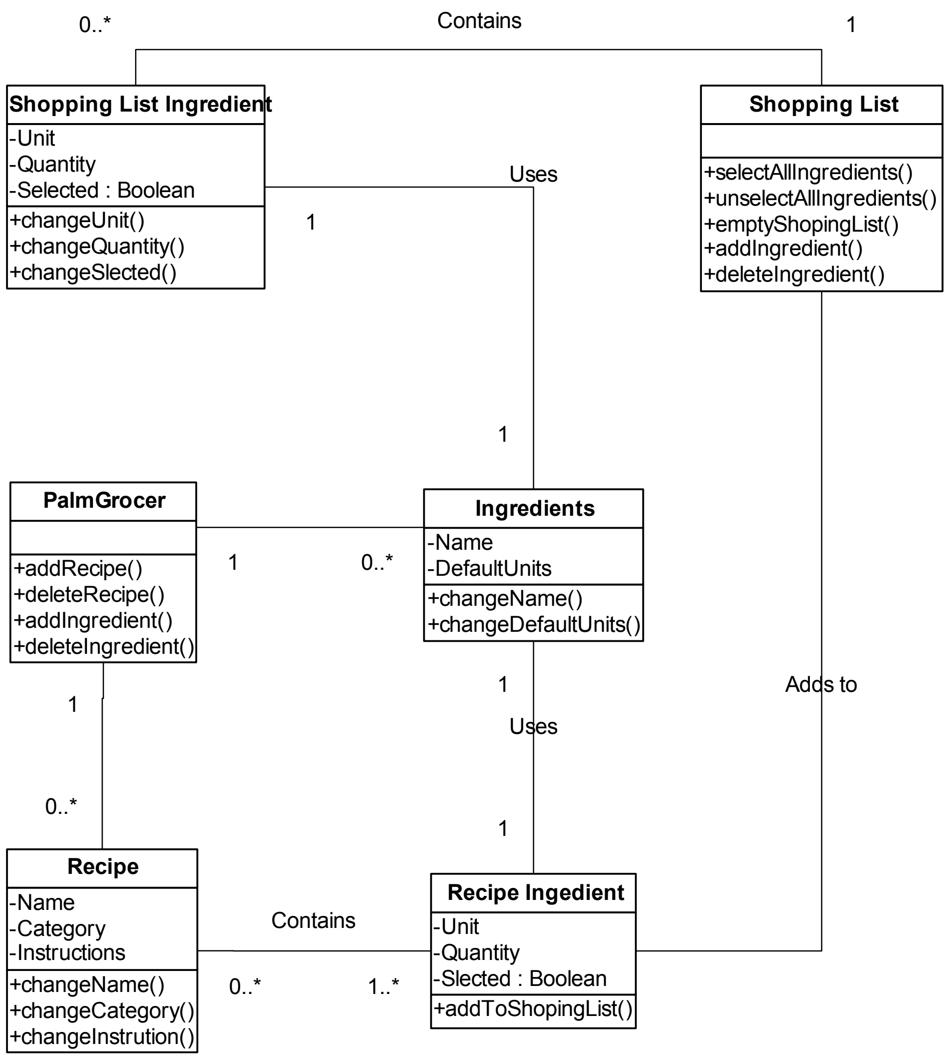
1 Domain Model

PalmGrocer domain model



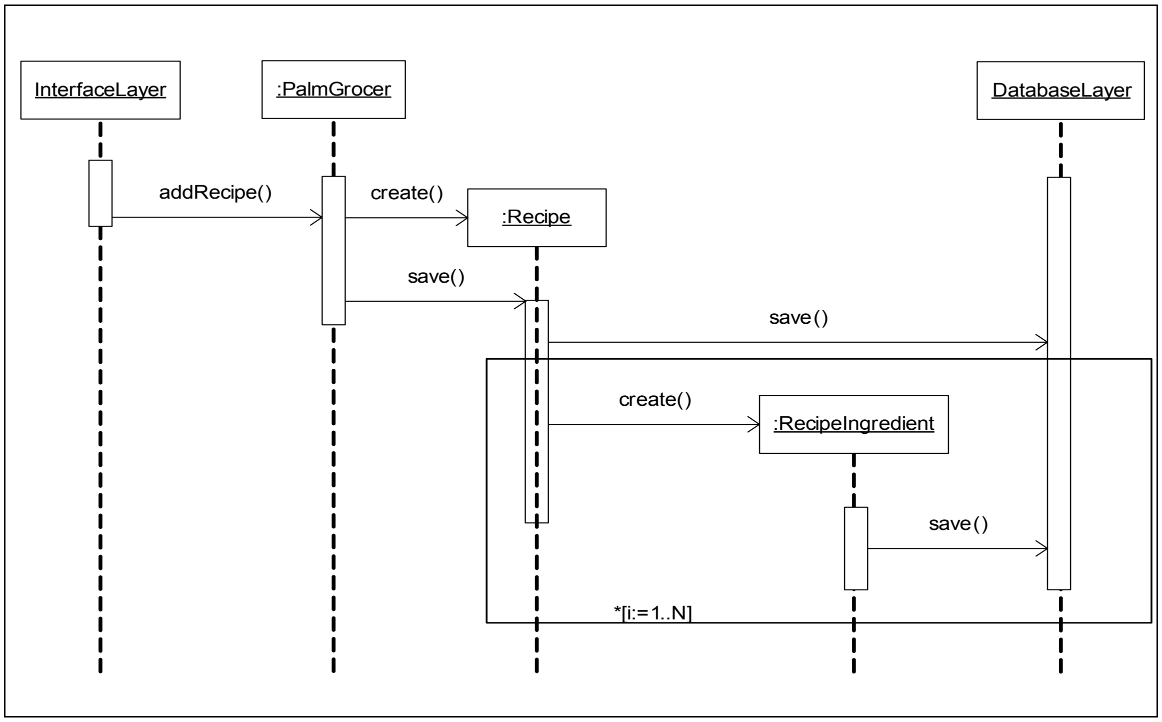
2 Design Model

2.3 Design Class Diagram

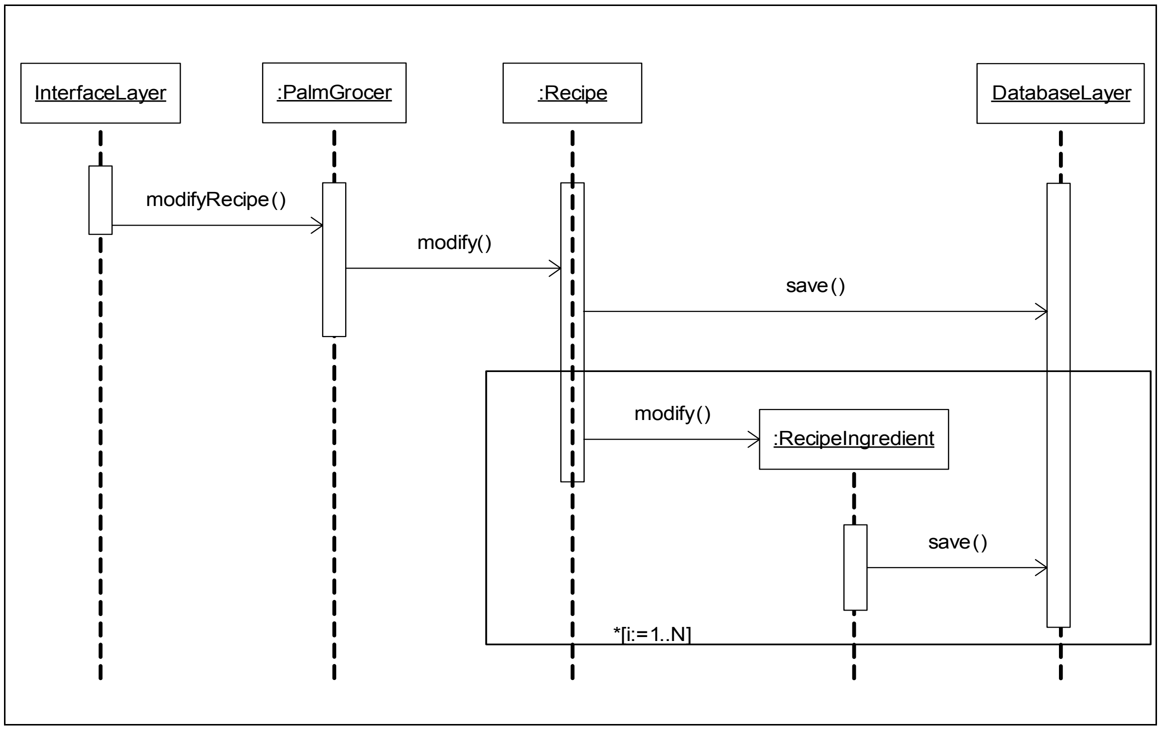


2.4 Sequence Diagrams

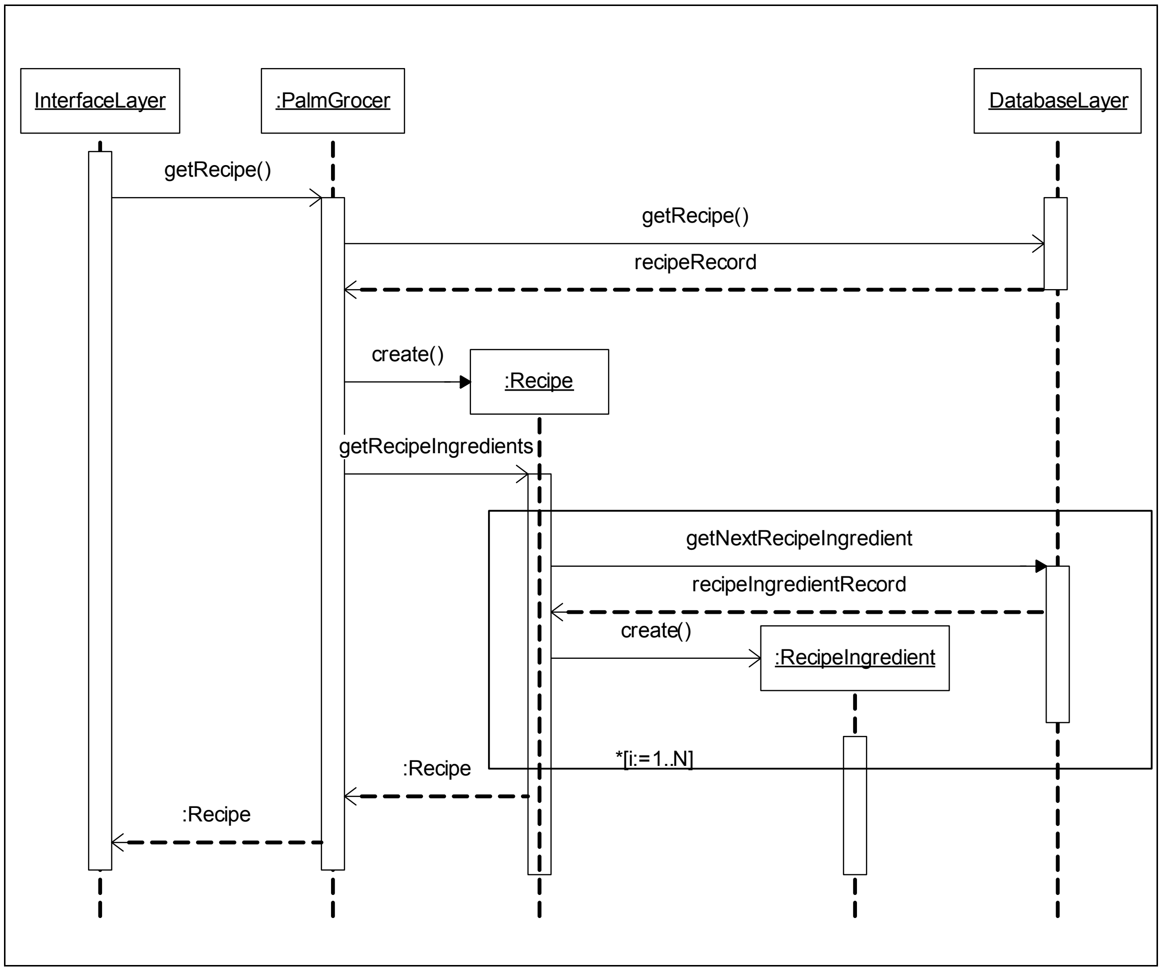
2.4.1 addRecipe



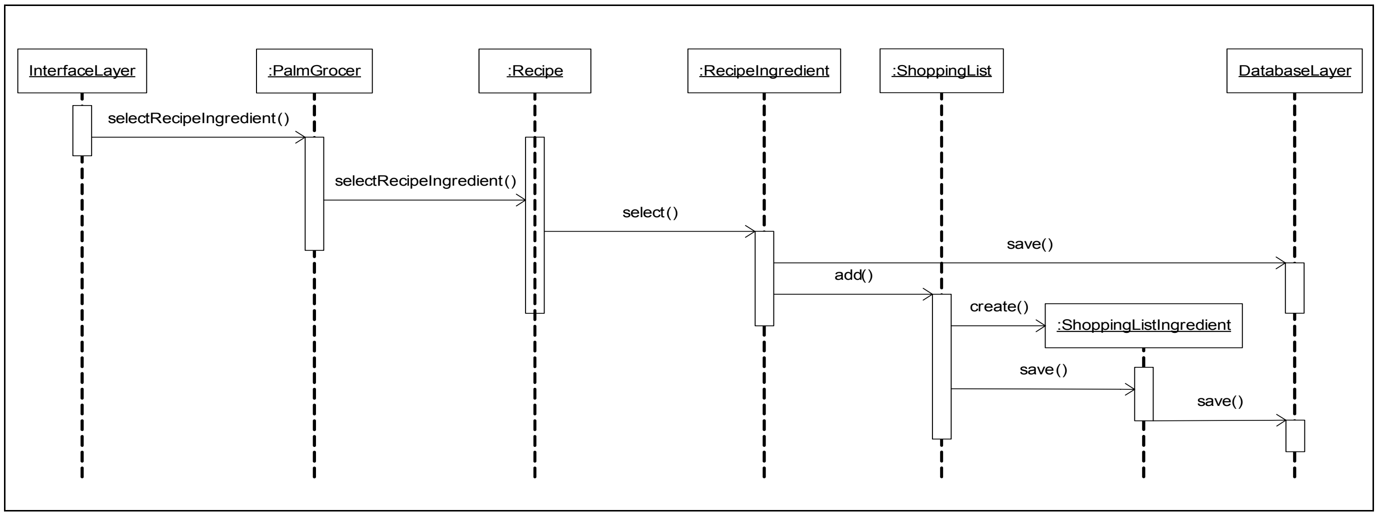
2.4.2 modifyRecipe



2.4.3 retrieveRecipe

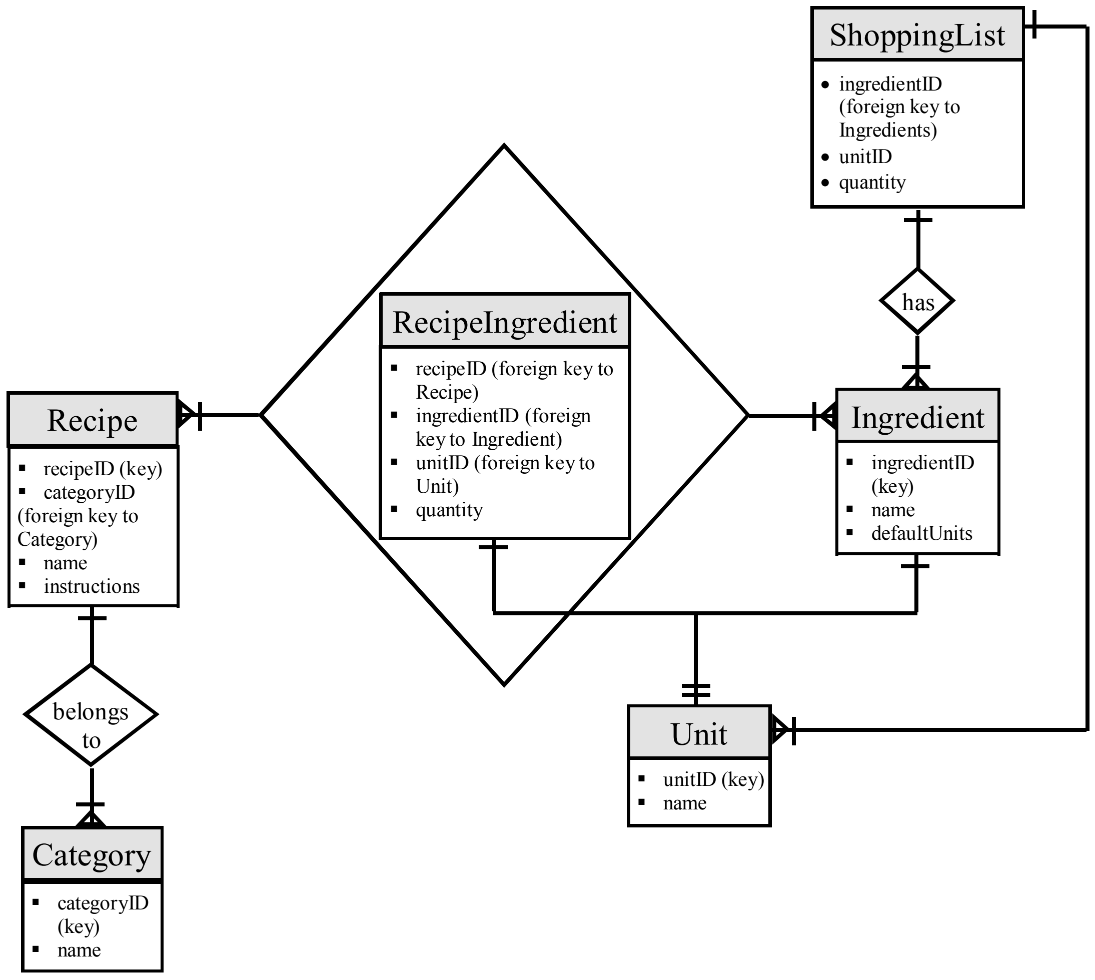


2.4.4 selectRecipeIngredient



3 Data Model

3.1 E-R Diagram



3.2 Data Dictionary

Recipe		
Attribute	Type	Description
recipeID	int	primary key
categoryID	int	foreign key to Category table
name	string	used to store the recipe name
instructions	string	step by step directions to make a recipe

RecipeIngredient		
Attribute	Type	Description
recipeID	int	foreign key to Recipe

ingredientID	int	foreign key to Ingredient
unitID	string	foreign key to Units
quantity	float	the amount of units needed

Ingredients		
Attribute	Type	Description
ingredientID	int	primary key
name	string	name of the ingredient
unitID	string	foreign key to Units - defaultUnits

Category		
Attribute	Type	Description
categoryID	int	primary key
name	string	name of the category

ShoppingList		
Attribute	Type	Description
ingredientID	int	foreign key to Ingredients
unitID	string	foreign key to Units
quantity	float	the amount of units needed
markedForDeletion	Boolean	

Units		
Attribute	Type	Description
unitID	int	primary key
name	string	name of the unit e.g. cup, teaspoon etc.

4 Test Model

4.1 Classes of tests

- Display of individual recipes
- Adding, changing and removing of recipes from database
- Exporting of recipe (and ingredients included) to shopping list
- Display of table of contents
- Display of shopping list
- Adding, removing and retrieving info from ingredients on shopping list
- Adding, removing and retrieving info from the master list
- Synching of Palm software with software on the PC
- Error handling

4.2 Expected software response

- User is able to add, edit and remove recipes from database
- User is able to specify which recipes are exported to shopping list, and which ingredients are included in the export
- User is able to view all the recipes in a list (Table of Contents), or limit that view to certain categories or other search criteria
- User is able to view all the ingredients in a shopping list, or limit that view to certain categories
- User is able to view information about ingredients on a shopping list, including which recipes – if any – have that ingredient in common
- User is able to remove ingredients from the shopping list by unchecking its checkbox
- System prevents user from deleting an ingredient or unit that is in use in a recipe, and displays an appropriate message.
- User is able to synch up the data in the PalmGrocer on their Palm device with the data on their PC.
- Errors handled gracefully; application does not “crash”.

4.3 Performance bounds

- Major functionality should be accessible with a minimum number of taps of the stylus, and a minimum amount of data entry on the device
- All major functionality should be available from the Palm and the PC, particularly data entry, which while time-consuming on the Palm, is much easier on a PC, which can then synch with a Palm.

4.4 Identification of critical components

Communication between main screen and shopping list is essential. Shopping list items can refer to the same ingredient from multiple recipes.

5 Implementation Model: Source Code

5.1 File: PalmGrocer.java

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import java.io.*;
import java.util.Vector;
import javax.microedition.rms.*;

public class PalmGrocer extends MIDlet
implements CommandListener
{
	Display display;

	private ChoiceGroup recipesMenu =
	    new ChoiceGroup( "Recipes", Choice.EXCLUSIVE );

	FindForm scrnFinder = new FindForm( this );
	EditRecipeForm scrnEditRecipe = new EditRecipeForm( this );
	ViewRecipeForm scrnMain = new ViewRecipeForm( this ); //For Main screen
	TOCForm scrnTOC = new TOCForm( this );
	ShoppingListForm scrnSL = new ShoppingListForm (this);
	MenuForm scrnMenu = new MenuForm ( this );

	static ChoiceGroup category = new ChoiceGroup( "Category:", Choice.EXCLUSIVE
);
	private static void buildCategoriesChoiceGroup()
	{

		String[] catList = { "All", "Dessert", "Rice", "Entree", "Soup",
		    "Unfiled", "Edit categories..." };

		for ( int n =0; n < catList.length; n++ )
		{
			category.append( catList[n], null );
		}

	}


	public PalmGrocer()
	{
		buildCategoriesChoiceGroup();
		buildMenus();
	}

	public void startApp()
	{
		display = Display.getDisplay(this);
		display.setCurrent( scrnMenu );
	}

	public void destroyApp( boolean unconditional)
	{
	}

	public void pauseApp()
	{

	}

	public void commandAction( Command c, Displayable s )
	{

	}


	// -- public static methods
	public void sendAlert( String msg )
	{
	  sendAlert( "Alert", msg );
	}

	public void sendAlert( String title, String msg )
	{
	Alert alert = new Alert(title, msg, null, null);
	alert.setTimeout(Alert.FOREVER);
	display.setCurrent(alert);
	}


	// -- private methods
	private void buildMenus()
	{
		recipesMenu.append( "Add/Edit", null );
		recipesMenu.append( "Find", null );
	}

}

class FindForm extends Form
implements CommandListener
{
	PalmGrocer pg = null;

	private TextField nameLike = new TextField( "Name like:", "", 32, 0 );
	private TextField ingredientsIncluded = new TextField( "Ingredients:", "",
256, 0 );

	private Command findCmd = new Command( "Find", Command.SCREEN, 1 );
	private Command resetCmd = new Command( "Reset", Command.SCREEN, 2);

	FindForm(PalmGrocer pg)
	{
		super( "Find Recipe" );

		this.pg = pg;

		this.addCommand( findCmd );
		this.addCommand( resetCmd );
		this.setCommandListener( this );
```

```
            this.append( nameLike );
            this.append( ingredientsIncluded );
      }

      public void commandAction( Command c, Displayable s )
      {
            String label = c.getLabel();
            if ( label.equals( Constants.FIND_RESET ) )
            {
                  nameLike.setString("");
                  ingredientsIncluded.setString("");
            }
            else  // hit SEARCH button
            {
                  pg.display.setCurrent( pg.scrnTOC );
            }
      }
}


class EditRecipeForm extends Form
    implements CommandListener
{

      PalmGrocer pg;
      private TextField name = new TextField( "Name:", "", 32, 0 );
      private TextField ingredients = new TextField( "Ingredients:", "", 256, 0 );
      private TextField instructions = new TextField( "Instructions:", "", 512, 0
);

      private Command okCmd = new Command( "OK", Command.SCREEN, 1 );
      private Command cancelCmd = new Command( "Cancel", Command.SCREEN, 2);

      EditRecipeForm( PalmGrocer pg )
      {
            super( "Edit Recipe" );
            this.pg = pg;
            this.addCommand( okCmd );
            this.addCommand( cancelCmd );
            this.setCommandListener( this );

            this.append( pg.category );
            this.append( name );
            this.append( ingredients );
            this.append( instructions );

      }

      public void commandAction( Command c, Displayable s )
      {
            String label = c.getLabel();

            if ( label.equals( Constants.OK ) )
            {
                //TODO: pg.scrnMain.gotoLatestRecord();
            }
```

```
			// whether user OK's or cancels, always return to view a recipe
			pg.display.setCurrent( pg.scrnMain );

		}

}

class ViewRecipeForm extends Form
implements CommandListener
{
		PalmGrocer pg;
		static private String LEMON_RICE_INSTR =
		    "1. Prepare rice.\n" +
		    "2. Allow rice to cool - spread grains\n" +
		    "3. Heat oil in pan, add mustard...";

		private TextField name = new TextField( "Name:", "Lemon Rice", 32, 0);
		private ChoiceGroup ing = new ChoiceGroup( "Ingredients:", Choice.MULTIPLE );
		private TextField instructions = new TextField( "Instructions:",
LEMON_RICE_INSTR, 512, 0 );

		private Command cmdPrev = new Command( Constants.MAIN_PREV, Command.SCREEN, 1
);
		private Command cmdNext = new Command( Constants.MAIN_NEXT, Command.SCREEN, 2
);
		private Command cmdEdit = new Command( Constants.MAIN_EDIT, Command.SCREEN, 3
);
		private Command cmdDel = new Command( Constants.MAIN_DELETE, Command.SCREEN,
4 );


		public ViewRecipeForm( PalmGrocer pg )
		{
			super( "Recipe Viewer" );
		    this.pg = pg;
			buildIngredientsChoiceGroup();

			this.append( name );
			this.append( ing );
			this.append( instructions );

			this.addCommand( cmdPrev );
			this.addCommand( cmdNext );
			this.addCommand( cmdEdit );
			this.addCommand( cmdDel );

			this.setCommandListener( this );

		}

		public void commandAction( Command cmd, Displayable d )
		{
		     if ( cmd == cmdEdit )
		     {
			    pg.display.setCurrent( pg.scrnEditRecipe );
		     }
```

```
          else if ( cmd == cmdNext )
          {
               pg.sendAlert( "STUB: Move to next record!" );
          }
          else if ( cmd == cmdPrev )
          {
               pg.sendAlert( "STUB: Move to previous record!" );
          }
          else if ( cmd == cmdDel )
          {
               pg.sendAlert( "STUB: delete this record!" );
          }

     }


     private void buildIngredientsChoiceGroup()
     {
           String[] ingList = { "Rice", "Lemon", "Mustard", "Split Peas",
           "Urad Dal", "Peanuts", "Green chili", "Turmeric", "Oil" };

           for ( int n =0; n < ingList.length; n++ )
           {
                 ing.append( ingList[n], null );
           }
     }
}

class ShoppingListForm extends Form
{

     PalmGrocer pg = null;
     private ChoiceGroup sl = new ChoiceGroup( "Groceries:", Choice.MULTIPLE );
     private Command cmdAdd = new Command( "Add", Command.SCREEN, 1 );
     private Command cmdDel = new Command( "Delete", Command.SCREEN, 2 );

     public ShoppingListForm(PalmGrocer pg)
     {
           super( "Shopping List" );
         this.pg = pg;
           buildSL();

           this.append( sl );
           this.addCommand( cmdAdd );
         this.addCommand( cmdDel );
     }

     private void buildSL()
     {
           String[] ingList = { "Rice", "Lemon", "Mustard", "Split Peas",
           "Urad Dal", "Peanuts", "Green chili", "Turmeric", "Oil", "Vodka Sauce
(Victoria Special)" };

           for ( int n =0; n < ingList.length; n++ )
           {
                 sl.append( ingList[n], null );
           }
```

```
		}
}//END ShoppingListForm


class TOCForm extends List
implements CommandListener
{
	PalmGrocer pg;

	private Command findCmd = new Command( Constants.TOC_SEARCH, Command.SCREEN, 1
);
	private Command filterCmd = new Command( Constants.TOC_VIEW_ALL,
Command.SCREEN, 2 );

	TOCForm( PalmGrocer pg )
	 {
		super( "Recipes", Choice.IMPLICIT );

		this.pg = pg;

		buildRecipesList();

		registerCommandListeners();

		this.addCommand( findCmd );
		this.addCommand( filterCmd );
	 }

	 public void commandAction( Command cmd, Displayable d )
	 {
		if ( cmd == findCmd )
		{
			pg.display.setCurrent( pg.scrnFinder );
		}
	 }



	 private void registerCommandListeners()
	 {
		this.setCommandListener( this );

	 }

	 private void buildRecipesList()
	 {
		 String[] recList = { "Rasmalai", "Apple Pie", "Lemon Rice",
			"Arroz con pollo", "Chicken noodle soup",
			"Aloo Gobi" };

		 for ( int n = 0; n < recList.length; n++ )
		 {
			 this.append( recList[n], null );
		 }
	 }
}
```

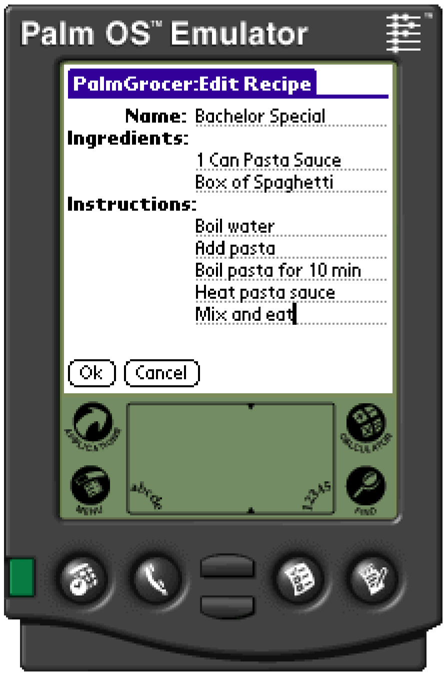
```
class MenuForm extends List
implements CommandListener
{
     PalmGrocer pg;


     MenuForm(PalmGrocer pg )
      {
           super( "Recipes", Choice.IMPLICIT );

           this.pg = pg;

           buildMenuList();

           registerCommandListeners();
      }

      public void commandAction( Command cmd, Displayable d )
      {
                List down = (List)pg.display.getCurrent();
                Screen curr = null;
                int index = down.getSelectedIndex();
                switch ( index )
                {
                    case 0: curr = pg.scrnMain; break;
                    case 1: curr = pg.scrnTOC;  break;
                    case 2: curr = pg.scrnSL;   break;
                    default: pg.sendAlert( "We cannot help you!  Index=" + index );
                }

                pg.display.setCurrent( curr );
      }

      private void registerCommandListeners()
      {
           this.setCommandListener( this );

      }

      private void buildMenuList()
      {
            String[] recList = { Constants.MENU_RECIPE_VIEWER,
Constants.MENU_LIST_RECIPES, Constants.MENU_SHOPPING_LIST};

            for ( int n = 0; n < recList.length; n++ )
            {
                  this.append( recList[n], null );
            }
      }

}//END MenuForm

class Constants
{
     public static final String OK = "OK";
     public static final String CANCEL = "Cancel";
     public static final String MENU_RECIPE_VIEWER = "Recipe Viewer";
     public static final String MENU_LIST_RECIPES = "List Recipes";
     public static final String MENU_SHOPPING_LIST = "Shopping List";

     public static final String FIND_SEARCH = "Search";
     public static final String FIND_RESET  = "Reset";

     public static final String MAIN_PREV = "<";
     public static final String MAIN_NEXT = ">";
     public static final String MAIN_EDIT = "Edit";

     public static final String MAIN_DELETE = "Del",
                                TOC_VIEW_ALL = "View All",
                                TOC_SEARCH = "Search";

      }
```

5.2 File: Recipes.java

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import java.io.*;
import javax.microedition.rms.*

public class Recipes extends MIDlet
                     implements CommandListener
{

    private Display display;
    private Alert alert;
    private Form newRecipe;
    private Form recipeDetail;

    private Command exitCmd;
    private Command gotoCmd;
    private Command cmdStore;

    private RecordStore rs = null;
    private RecordEnumeration re = null;

    TextField name = null;
    ChoiceGroup category = null;
    ChoiceGroup ing = null;


    public Recipes()
    {
        newRecipe = new Form( "Recipe Form" );
        name = new TextField( "Name:", "", 32, 0 );
        category = new ChoiceGroup( "Category", Choice.EXCLUSIVE );
        ing = new ChoiceGroup( "Ingredients", Choice.MULTIPLE );

        gotoCmd = new Command( "Goto Detail", Command.SCREEN, 1 );
        cmdStore = new Command( "Store data", Command.SCREEN, 2);
        newRecipe.addCommand( gotoCmd );
        newRecipe.addCommand( cmdStore );
        newRecipe.setCommandListener( this );

        recipeDetail = new Form( "Recipe Detail" );
        exitCmd = new Command( "End It All", Command.SCREEN, 1 );
        recipeDetail.addCommand( exitCmd );
        recipeDetail.setCommandListener( this );
    }

    public void startApp()
    {
        display = Display.getDisplay(this);
        category.append( "French", null );
        category.append( "Indian", null );
        category.append( "Italian", null );
        category.append( "Spanish", null );
        category.append( "Thai", null );
        category.append( "American", null );

        ing.append( "Rice", null );
        ing.append( "Dahl", null );

        newRecipe.append( name );
        newRecipe.append( category );
        newRecipe.append( ing );

        display.setCurrent( newRecipe );
    }


    public void commandAction( Command c, Displayable s )
    {

        if ( c == exitCmd )
        {
            destroyApp(false);
            notifyDestroyed();
        }
        else if ( c == gotoCmd )
        {
            display.setCurrent( recipeDetail );
        }
        else if ( c == cmdStore )
        {
            try
            {
                  //alert = new Alert("cmdStore ", "Storing...  ",
null, AlertType.INFO);
                  //alert.setTimeout(Alert.FOREVER);
                  //display.setCurrent(alert);
                  rs = RecordStore.openRecordStore("recipeRS", true);

                  alert = new Alert("rs " + rs);
                  alert.setTimeout(Alert.FOREVER);
                  display.setCurrent(alert);
            }
            catch (Exception error)
            {
                  alert = new Alert("Blah");
                  alert.setTimeout(Alert.FOREVER);
                  display.setCurrent(alert);

            }
        }
        else
        {

        }
    }


    public void pauseApp()
    {

    }

    public void destroyApp( boolean unconditional )
    {

    }

}
```

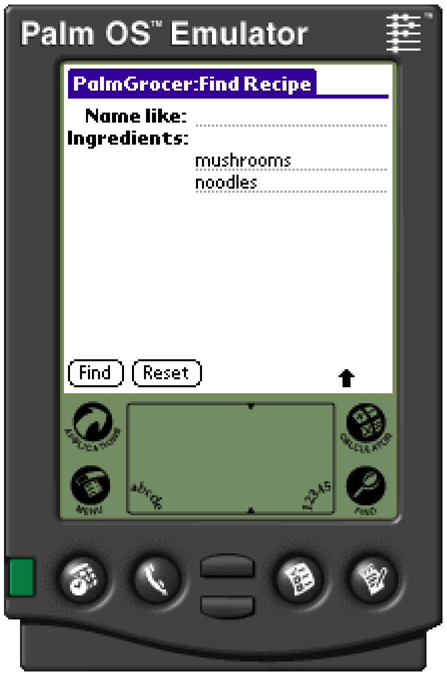
6 User Interface

6.1 Edit/ Add Recipe



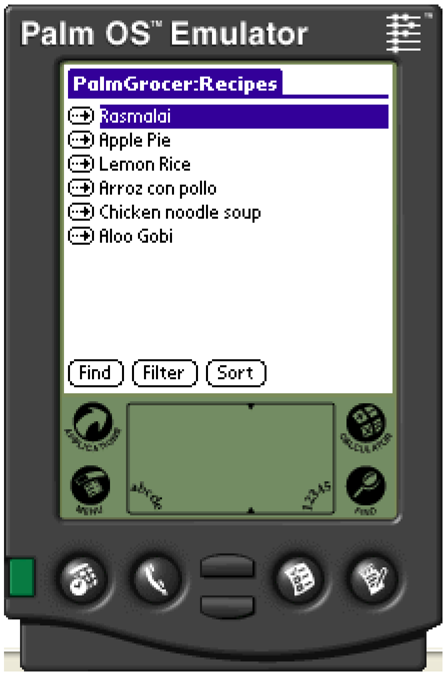
Name: Bachelor Special
Ingredients:
1 Can Pasta Sauce
Box of Spaghetti
Instructions:
Boil water
Add pasta
Boil pasta for 10 min
Heat pasta sauce
Mix and eat

6.2 Find Recipe



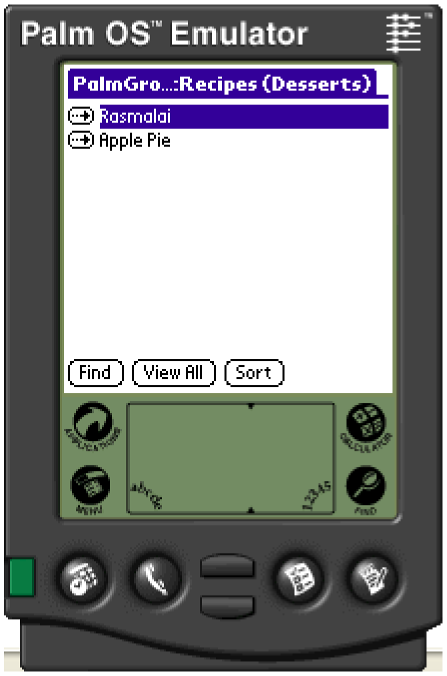
Name like:
Ingredients:
mushrooms
noodles

6.3 Table of Contents-- Unfiltered



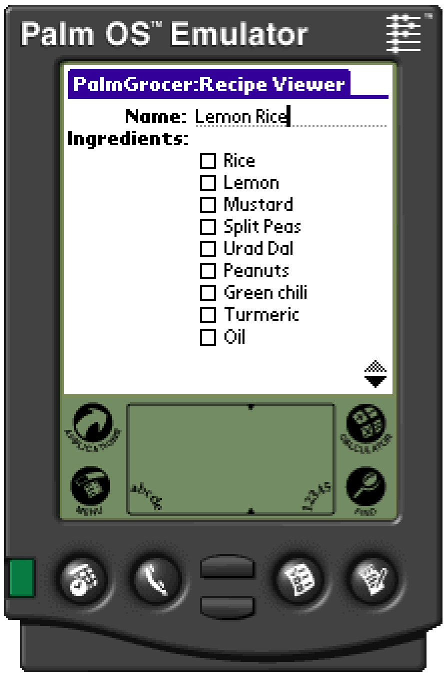
Rasmalai
Apple Pie
Lemon Rice
Arroz con pollo
Chicken noodle soup
Aloo Gobi

6.4 Table of Contents--Filtered



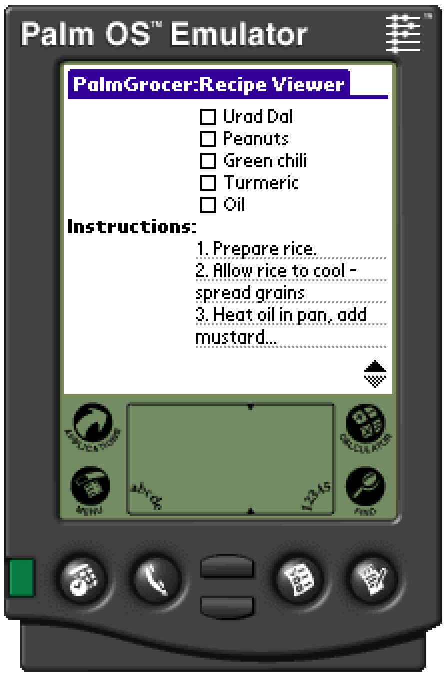
Rasmalai
Apple Pie

6.5 View Recipe Form—Top



Name: Lemon Rice
Ingredients:
- ☐ Rice
- ☐ Lemon
- ☐ Mustard
- ☐ Split Peas
- ☐ Urad Dal
- ☐ Peanuts
- ☐ Green chili
- ☐ Turmeric
- ☐ Oil

6.6 View Recipe Form—Bottom



- ☐ Urad Dal
- ☐ Peanuts
- ☐ Green chili
- ☐ Turmeric
- ☐ Oil

Instructions:
1. Prepare rice.
2. Allow rice to cool - spread grains
3. Heat oil in pan, add mustard...