Payment Using Loyalty Points Pseudo-code

```
class LoyaltyPoints implements Payment {


  private int balance;
  private final int DOLLAR_TO_POINTS = 100;

  public boolean charge(double amount) {
    int pointAmount  = toPoints(amount);

    if (amount > balance) {
      throw new InsufficientFundsException();
    }

    balance -= amount;

    return true;
  }

  public boolean refund(double amount) {
    int pointAmount  = toPoints(amount);

    balance += pointAmount;
    return true;

  }

  public int updateBalance(double amount) {
    int pointAmount  = toPoints(amount);
    balance += pointAmount;
    return pointAmount;
  }

  public int getBalance() {
    return balance;
  }


  private int toPoints(double amount) {
    return (int) (amount * DOLLAR_TO_POINTS);
  }



}
```

Filtering Menus based on Dietary Constraints

```
class Menu {

    List<FoodItem> foodList;

    public Menu(List<FoodItem> foodList) {
        this.foodList = foodList;
    }

    public List<FoodItem> filterMenuByDiet (DietType filteredDiet) {
        List<FoodItem> filteredList = new ArrayList<FoodItem>();
        for (FoodItem foodItem : this.foodList) {
            List<Ingredient> ingredientList = foodItem.getIngredients();
            for (Ingredient ingredient : ingredientList) {
                if (!ingredient.violatesDiet(filteredDiet)) {
                    filteredList.add(foodItem);
                }
            }
     }

        return filteredList;

    }

    public List<FoodItem> filterMenuByAllergens(List<Ingredient> customerAllergens) {
        List<FoodItem> filteredList = new ArrayList<FoodItem>();
        for (FoodItem foodItem : this.foodList) {
            for (Ingredient ingredient : customerAllergens) {
                if (!foodItem.getIngredients().contains(ingredient)) {
                    filteredList.add(foodItem);
                }
            }

        }

        return filteredList;
    }

}
```

Ordering a Previous or Favorite Order

```
  public boolean addPrevOrder(Order old_order) {
    savedOrders.add(old_order);
    return true;
  }

  public boolean reOrderPrev(int orderID) {
    Order prev_order = savedOrders.get(orderID);
    if (!addToCart(prev_order)) {
      throw new ItemsUnavailableException();
    }
    return true;
  }

  public boolean addToCart(Order old_order) {
    for (FoodItem item : old_order.items_list) {
      if (!item.isAvailable) {
        return false;
      }
      getUser().addItem(item);
    }
  }

}
```

Logging a User into the Application

```
    ArrayList<User> users = new ArrayList<User>();

    public boolean login() {
        Scanner keyboard = new Scanner(System.in);

        // Gets input from users.
        System.out.println("Enter Username: ");
        String username = keyboard.nextLine();
        System.out.println("Enter Password: ");
        String password = keyboard.nextLine();

        boolean login = false;

        for (User temp : users) {
            if (username.equals(temp.getUser()) && password.equals(temp.getPass())) {
                System.out.println("Welcome " + temp.getUser() + " to truckd.");
                login = true;
                break;
            } else {
                login = false;
            }
        }

        if (login == false) {
            System.out.println("Username or password is incorrect. Please try again.");
        }

        return login;
    }

    public void saveUser(User user) {
        users.add(user);
    }

    public ArrayList<User> getUsers() {
        return users;
    }

}
```

Applying promotion/coupon codes to an order

```
    public Promo isValidCode(String code) {
        Promo ret = getPromo(code);
        if (ret == NULL) {
            throw new InvalidPromoCodeException();
        }
        return ret;
    }

    public double getDiscount(String code) {
        Promo ret = isValidCode(code);
        return ret.discount;
    }

    public void updateOrderTotal(Order ord, String code) {
      double disc = getDiscount(code);
      double newTotal = ord.total * (1 - disc);
      ord.updateTotal(newTotal);
    }

    public double displaySavings(Order, String code) {
      double disc = getDiscount(code);
      double newTotal = ord.total * (1 - disc);
      return ord.total - newTotal;
    }
}
```