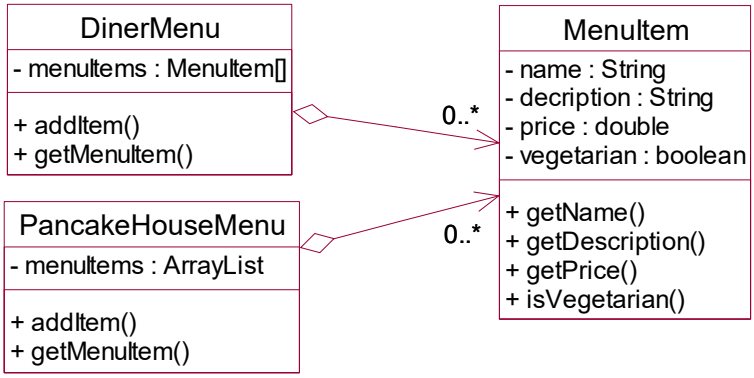
ITERATOR PATTERN

Problem1:

1. We have two menus: Dinner's menu and Pancake House's menu. Dinner's menu uses an Array to hold his menu's items. Otherwise, Pancake House's menu uses an ArrayList. Now, we want to get both menus in only one place.



2. Create a Java project called ***RestaurantMenu1*** and implements these specifies:
   + The **`MenuItem`** class has these attributes: **`name`**, **`price`**, **`description`** and a **`flag`** to indicate if the item is vegetable. It also has the getter method to access these fields.

```
public class MenuItem {
   String name;
   String description;
   boolean vegetarian;
   double price;

   public MenuItem(String name, String description, boolean vegetarian,
         double price) {
      this.name = name;
      this.description = description;
      this.vegetarian = vegetarian;
      this.price = price;
   }
   // your code here
}
```

   + You have to create **`PancakeHouseMenu`** class to implement the Pancake House menu. Here, you'll use **`ArrayList`** to store the menu items and apply **`addItem()`** method to add a new item to the **`ArrayList`**. The **`getMenuItem()`** method returns the list of menu items, you also provide other methods if you want.

```
public class PancakeHouseMenu {
   ArrayList menuItems;

   public PancakeHouseMenu() {
      menuItems = new ArrayList();

      addItem("K&B's Pancake Breakfast",
            "Pancakes with scrambled eggs, and toast", true, 2.99);
      addItem("Regular Pancake Breakfast", "Pancakes with fried eggs, sausage",
            false, 2.99);
```

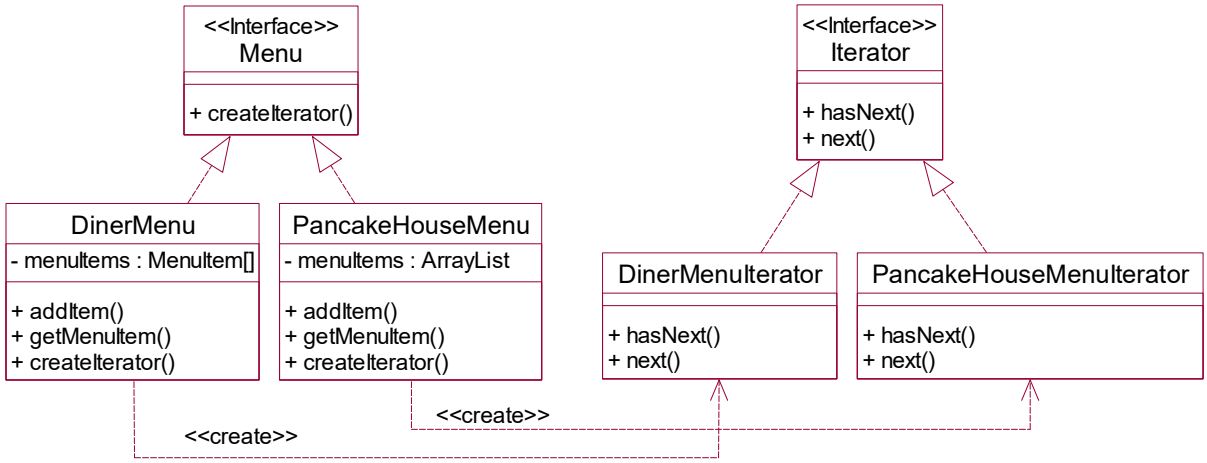
```
        addItem("Blueberry Pancakes", "Pancakes made with fresh blueberries",
              true, 3.49);
        addItem("Waffles", "Waffles, with your choice of blueberries or strawberries",
              true, 3.59);
    }

    public void addItem(String name, String description, boolean vegetarian,
          double price) {
        // your code here
    }

    public ArrayList getMenuItems() {
        // your code here
    }

    public String toString() {
        return "Objectville Pancake House Menu";
    }
}
```

+ The **`DinnerMenu`** class has three attributes: the **max size of the menu**, **number of items**, and an **`Array`** to store the menu items. The **`addItem()`** method is also created, it takes all necessary parameter to create a **`MenuItem`** and instate one. You must sure that you aren't out of the menu items' limit size before you add a new item. The **`getMenuItem()`** method returns an array of menu items that you have. You can implement other methods yourself if you want.

```
public class DinerMenu {
    static final int MAX_ITEMS = 6;
    int numberOfItems = 0;
    MenuItem[] menuItems;

    public DinerMenu() {
        menuItems = new MenuItem[MAX_ITEMS];
        addItem("Vegetarian BLT",
            "(Fakin') Bacon with lettuce & tomato on whole wheat", true, 2.99);
        addItem("BLT", "Bacon with lettuce & tomato on whole wheat", false, 2.99);
        addItem("Soup of the day",
              "Soup of the day, with a side of potato salad", false, 3.29);
        addItem("Hotdog","A hot dog, with saurkraut, relish, onions, topped with cheese",
               false, 3.05);
        addItem("Steamed Veggies and Brown Rice", "Steamed vegetables over brown rice",
              true, 3.99);
        addItem("Pasta", "Spaghetti with Marinara Sauce, and a slice of sourdough bread",
              true, 3.89);
    }

    public void addItem(String name, String description, boolean vegetarian,
          double price) {
        // your code here
    }

    public MenuItem[] getMenuItems() {
        // your code here
    }
}
```

+ The **`getMenuItem()`** method of **`PancakeHouseMenu`** and **`DinnerMenu`** class looks the same, but they returns difference types. If you want to print out the items from **`PancakeHouseMenu`** you have to loop through an **`ArrayList`**, and if you want to print out the **`DinnerMenu`** you have to loop through an **`Array`**. It's bore. How can we solve these problems?

  Solution: using **Iterator Pattern**



+ Create **`Iterator`** interface with methods: **`hasNext()`** tells us if there are more element in the aggregate to iterate through, **`next()`** returns the next object of the aggregate.

```
public interface Iterator {
   boolean hasNext();
   Object next();
}
```

+ Create the **`Menu`** interface, which is common interface for **`PancakeHouseMenu`** and **`DinnerMenu`** class. It specifies the new method, **`createIterator()`** that create **`Iterator`** for menu.

```
public interface Menu {
   public Iterator createIterator();
}
```

+ Create **`DinnerMenuIterator`** class, which is an implementation of **`Iterator`** that knows how to iterate over an Array of **`MenuItem`**. This class has two attribute: An Array of **`MenuItem`** to store all menu items. Current position of the iteration over the array.
+ The **`next()`** method return the next item of the menu, remember to increase the current position. The **`hasNext()`** method return a boolean indicating whether or not there are more element in the menu items. It's also check if the next item is null.

```
public class DinerMenuIterator implements Iterator {
   MenuItem[] items;
   int position = 0;

   public DinerMenuIterator(MenuItem[] items) {
      this.items = items;
   }
```

```
    public Object next() {
        // your code here
    }

    public boolean hasNext() {
        // your code here
    }
}
```

+ Now, your **DinnerMenu** implement the **Menu** interface, that need to implement the **createIterator()** method to create **DinerMenuIterator** object.

```
public class DinerMenu implements Menu {
    // the same old codes

    public Iterator createIterator() {
      // your code here
    }
}
```

+ You can also create **PancakeHouseMenuIterator** to iterate over an **ArrayList** of **MenuItem**

```
public class PancakeHouseMenuIterator implements Iterator {
    ArrayList items;
    int position = 0;

    public PancakeHouseMenuIterator(ArrayList items) {
        this.items = items;
    }

    public Object next() {
        // your code here
    }

    public boolean hasNext() {
        // your code here
    }
}
```

+ Amend the **PancakeHouseMenu** similar **DinnerMenu**.

```
public class PancakeHouseMenu implements Menu {
    // the same old codes

    public Iterator createIterator() {
      // your code here
    }
}
```

+ **Waitress** class has two menu items: **PancakeHouseMenu** and **DinnerMenu**. In **Waitress** class you must implement these method:
    - **printMenu()** method prints every item in the menu
    - **printVegetarianMenu()** prints all vegetarian menu items

- In the **`isItemVegerian(String name)`** method you have to give the name of an item, it returns true if the item is vegetarian, otherwise return false.

```
public class Waitress {
   PancakeHouseMenu pancakeHouseMenu;
   DinerMenu dinerMenu;

   public Waitress(PancakeHouseMenu pancakeHouseMenu, DinerMenu dinerMenu) {
      this.pancakeHouseMenu = pancakeHouseMenu;
      this.dinerMenu = dinerMenu;
   }

   public void printMenu() {
      Iterator pancakeIterator = pancakeHouseMenu.createIterator();
      Iterator dinerIterator = dinerMenu.createIterator();
      System.out.println("MENU\n----\nBREAKFAST");
      printMenu(pancakeIterator);
      System.out.println("\nLUNCH");
      printMenu(dinerIterator);
   }

   private void printMenu(Iterator iterator) {
      // your code here
   }

   public void printVegetarianMenu() {
      printVegetarianMenu(pancakeHouseMenu.createIterator());
      printVegetarianMenu(dinerMenu.createIterator());
   }

   private void printVegetarianMenu(Iterator iterator) {
      // your code here
   }

   public boolean isItemVegetarian(String name) {
      Iterator breakfastIterator = pancakeHouseMenu.createIterator();
      if (isVegetarian(name, breakfastIterator)) {
         return true;
      }
      Iterator dinnerIterator = dinerMenu.createIterator();
      if (isVegetarian(name, dinnerIterator)) {
         return true;
      }
      return false;
   }


   private boolean isVegetarian(String name, Iterator iterator) {
      // your code here
   }
}
```

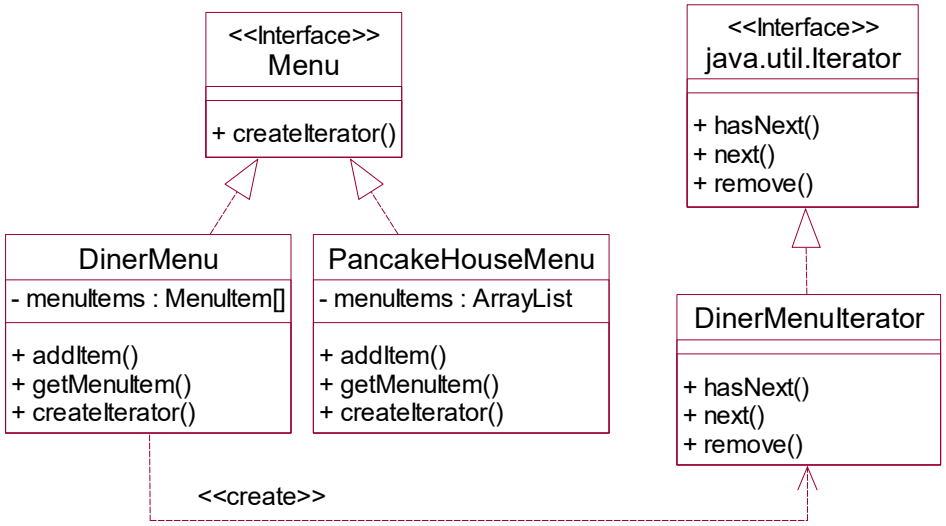
+ Create the **`MenuTestDrive`** to test your implementation.

```
public class MenuTestDrive {
   public static void main(String args[]) {
      PancakeHouseMenu pancakeHouseMenu = new PancakeHouseMenu();
```

```
        DinerMenu dinerMenu = new DinerMenu();
        Waitress waitress = new Waitress(pancakeHouseMenu, dinerMenu);
        waitress.printMenu();
        waitress.printVegetarianMenu();

        System.out.println("\nCustomer asks, is the Hotdog vegetarian?");
        System.out.print("Waitress says: ");
        if (waitress.isItemVegetarian("Hotdog")) {
            System.out.println("Yes");
        } else {
            System.out.println("No");
        }
        System.out.println("\nCustomer asks, are the Waffles vegetarian?");
        System.out.print("Waitress says: ");
        if (waitress.isItemVegetarian("Waffles")) {
            System.out.println("Yes");
        } else {
            System.out.println("No");
        }

    }
}
```

Problem2:

1. In this solution, you have to define a common interface for **`PancakeHouseMenu`** and **`DinnerMenu`** and clean up the **`Waitress`** class little more. You should use the **Java Iterator interface** instead of your **`Iterator`**.



2. Create ***RestaurantMenu2*** project, you can copy **ReataurantMenu1** project and implement these changes
   + **`DinnerMenuIterator`** class now implement the **`java.util.Iterator`** interface so it has to override three methods**: `hasNext(), next(), remove().`**
     - When you implement the **`remove()`** method, you must sure that you don't access out of menu item's bound.

```
public class DinerMenuIterator implements java.util.Iterator {
   MenuItem[] list;
   int position = 0;

   public DinerMenuIterator(MenuItem[] list) {
      this.list = list;
   }

   public Object next() {
     // your code here
   }

   public boolean hasNext() {
     // your code here
   }

   public void remove() {
      if (position <= 0) {
        throw new IllegalStateException(
          "You can't remove an item until you've done at least one next()");
      }
      if (list[position - 1] != null) {
         for (int i = position - 1; i < (list.length - 1); i++) {
            list[i] = list[i + 1];
         }
```

```
            list[list.length - 1] = null;
        }
    }
}
```

+ **DinnerMenu** implement the **createIterator()** of the **Menu** interface that return **DinerMenuIterator** object.

```
public class DinerMenu implements Menu {
    // the same old codes

    public Iterator createIterator() {
      // your code here
    }
}
```

+ **PancakeHouseMenu** implement the **Menu** interface: Instead of create your own **Iterator**, now you just call the **iterator()** method on the menu items.

```
public class PancakeHouseMenu implements Menu {
    // the same old codes

    public Iterator createIterator() {
      // your code here
    }
}
```

+ The ***Waitress, MenuTestDrive*** is similar.

Problem3:

1. Now, we want you to add **CafeMenu** into your restaurant. Note that the **CafeMenu** stores its menu items in a **HashTable**.
2. Create another Java project called **RestaurantMenu3**, you can copy **ReataurantMenu2** project and implement these changes
   + Create **CafeMenu** class, which implement **Menu** interface. You can use **HashTable** or **HashMap** data structure to store your items. The **createIterator()** method to transfers the **Hashtable** or **HashMap** to **Iterator**, remember that we're not getting an Iterator for the whole **Hashtable** (**HashMap**), just for the values.

```
public class CafeMenu implements Menu {
    Hashtable menuItems = new Hashtable();

    public CafeMenu() {
        addItem("Veggie Burger and Air Fries",
              "Veggie burger on a whole wheat bun, lettuce, tomato, and fries",
               true, 3.99);
        addItem("Soup of the day",
               "A cup of the soup of the day, with a side salad", false, 3.69);
        addItem("Burrito",
             "A large burrito, with whole pinto beans, salsa, guacamole", true,
             4.29);
    }
```

```
    public void addItem(String name, String description, boolean vegetarian,
          double price) {
      // your code here
    }

    public Hashtable getItems() {
       return menuItems;
    }

    public Iterator createIterator() {
      // your code here
    }
}
```

+ You have to modify a little in the **`Waitress`** class combine all menu items so that you needn't change your code?
  - Now, it just takes an **`ArrayList`** of menu items.
  - In the **`printMenu()`** method, you'll iterate through the menu, passing each the menu's **`iterator`** to the overloaded **`printMenu()`** method.

```
public class Waitress {
   ArrayList menus;

   public Waitress(ArrayList menus) {
      this.menus = menus;
   }

   public void printMenu() {
     // your code here
   }

   void printMenu(Iterator iterator) {
     // your code here
   }
}
```

+ Write **`MenuTestDrive`** class to test your new implementations.