Implementing Encapsulation and Constructors

As we learned in Ch12 Classes Part II, we can do better control of class members to hide the sensitive data from users by using encapsulation.

Example

```
class Person
{
   private String name; // private = restricted access // field

   public String getName() { // get method
      return name;
   }

   public void setName(String name) { // set method
      this.name = name;
   }
}
```

The get methods will return the values of field variables in code if we create a constructor associated with parameters which correspond to the field variables.

Example

```
class Person
{
   private String name; // private = restricted access // field
   private int age; // private = restricted access // field

   public Person(String name, int age)
   {
      this.age = age;
      this.name = name;
   }

   public String getName() { // get method
      return name;
   }

   public int getAge() { // get method
```

```
        return age;
    }
}
```

The following example that we use encapsulation to protect data and create constructors to initialize objects.

Example

```
class Person
{
    private String name; // private = restricted access // field
    private int age; // private = restricted access // field

    // Create a class constructor with parameters
    public Person(String name, int age)
    {
        this.age = age;
        this.name = name;
    }

    public String getName() { // get method
        return name;
    }

    public int getAge() { // get method
        return age;
    }
}

class Program
{
    public static void main(String[] args)
    {
        Person person1 = new Person("John", 21);
        Person person2 = new Person("Mary", 20);
        Person person3 = new Person("Anderson", 22);
        Person person4 = new Person("Lucy", 21);
        Person person5 = new Person("Jenson", 23);
        Person person6 = new Person("Emily", 19);
```

```
        System.out.println(person1.getName() + " is " + person1.getAge() + " years old.");
        System.out.println(person2.getName() + " is " + person2.getAge() + " years old.");
        System.out.println(person3.getName() + " is " + person3.getAge() + " years old.");
        System.out.println(person4.getName() + " is " + person4.getAge() + " years old.");
        System.out.println(person5.getName() + " is " + person5.getAge() + " years old.");
        System.out.println(person6.getName() + " is " + person6.getAge() + " years old.");
    }
}
```

Objects in a List

The type parameter used in creating a list defines the type of the variables that are added to the list. For instance, List<String> includes Strings, List<Integer> integers, and List<Double> floating point numbers.

```
List<String> names = new ArrayList<String>();
```

Strings are objects, therefore the other kinds of objects can also be found in lists. In the example below we first add objects to a list, after which the objects in the list are printed one by one.

Example

```
import java.util.*;

class Person
{
    private String name; // private = restricted access // field
    private int age; // private = restricted access // field

    // Create a class constructor with parameters
    public Person(String name, int age)
    {
        this.age = age;
        this.name = name;
    }

    public String getName() { // get method
        return name;
    }
```

```
    public int getAge() { // get method
        return age;
    }
}

class Program
{
    public static void main(String[] args)
    {
        List<Person> persons = new ArrayList<Person>();
        persons.add(new Person("John", 21));
        persons.add(new Person("Mary", 20));
        persons.add(new Person("Anderson", 22));
        persons.add(new Person("Lucy", 21));
        persons.add(new Person("Jenson", 23));
        persons.add(new Person("Emily", 19));

        for (Person person: persons) {
            System.out.println(person.getName() + " is " + person.getAge() + " years old.");
        }
    }
}
```

The following example that we can add an user input to ask for age limit and search the requested persons.

Example

```
import java.util.*;

class Person
{
    private String name; // private = restricted access // field
    private int age; // private = restricted access // field

    // Create a class constructor with parameters
    public Person(String name, int age)
    {
        this.age = age;
        this.name = name;
    }
```

```
    public String getName() { // get method
        return name;
    }

    public int getAge() { // get method
        return age;
    }
}

class Program
{
    public static void main(String[] args)
    {
        List<Person> persons = new ArrayList<Person>();
        persons.add(new Person("John", 21));
        persons.add(new Person("Mary", 20));
        persons.add(new Person("Anderson", 22));
        persons.add(new Person("Lucy", 21));
        persons.add(new Person("Jenson", 23));
        persons.add(new Person("Emily", 19));

        // Creating Scanner class object(產生 Scanner 類別物件)
        Scanner scan = new Scanner(System.in);

        // Ask for age limit
        System.out.println("What is the age limit? ");
        int ageLimit = scan.nextInt();

        // Print only those who are above the limit
        for (Person person: persons) {
            if (person.getAge() >= ageLimit)
            {
                System.out.println(person.getName() + " is " + person.getAge() + " years old.");
            }
        }
    }
}
```

The following example that we use while loop to add objects to a list repeatedly, and then the objects in the list are printed one by one.

Example

```
import java.util.*;

class Person
{
   private String name; // private = restricted access // field
   private int age; // private = restricted access // field

   // Create a class constructor with parameters
   public Person(String name, int age)
   {
      this.age = age;
      this.name = name;
   }

   public String getName() { // get method
      return name;
   }

   public int getAge() { // get method
      return age;
   }
}

class Program
{
   public static void main(String[] args)
   {
      List<Person> persons = new ArrayList<Person>();

      // Read the names of persons from the user input
      while (true)
      {
         // Creating Scanner class object(產生 Scanner 類別物件)
         Scanner scan = new Scanner(System.in);

         System.out.println("Enter a name, empty will stop: ");
```

```
            String name = scan.nextLine(); if (name.equals("")) { break; }

            System.out.println("Enter the age of the person " + name + ": ");
            int age = scan.nextInt();

            // add to the list a new person and whose name is the previous user input
            persons.add(new Person(name, age));
        }
        // Print the number of the entered persons, and their individual information
        System.out.println();
        System.out.println("Persons in total: " + persons.size());
        System.out.println("Persons: ");

        for (Person person: persons)
        {
            System.out.println(person.getName() + " is " + person.getAge() + " years old.");
        }
    }
}
```