Data Abstraction
Review of Java Constructs

Janyl Jumadinova

September 9, 2020
An **array** is a logical, homogenous collection of data elements, grouped together under a common variable name and referenced individually by position number.
An **array** is a logical, homogenous collection of data elements, grouped together under a common variable name and referenced individually by position number.

An array in Java is a **reference type** since it is an object.
An **array** is a logical, homogenous collection of data elements, grouped together under a common variable name and referenced individually by position number.

An array in Java is a **reference type** since it is an object.

To refer to a particular location or element in the array, we specify the name of the array and the position number of the particular element in the array.
int [ ] numbers = new int[10];

- Arrays occupy a fixed amount of space in memory.
- The programmer specifies the type of each element and the number of elements required by each array so that the compiler may reserve the appropriate amount of space in memory for the array.
What is an Object?

- An object is any item that can be described by listing a set of attributes or characteristics, and a set of behaviors or actions.
- The behaviors typically involve, in some fashion, the attributes, either by using their values, examining their values or modifying their values.
- Attributes/characteristics are also called the data members of the object.
- Behaviors are also called the member functions or methods of the object.
What is a Class?

A class is a template for an object.

- A class specifies the set of characteristics and the set of behaviors that an object will have.
What is a Class?

A class is a template for an object.

- A class specifies the set of characteristics and the set of behaviors that an object will have.
- Any object of a particular class will have the same set of characteristics and behaviors.
What is a Class?

A class is a template for an object.

- A class specifies the set of characteristics and the set of behaviors that an object will have.
- Any object of a particular class will have the same set of characteristics and behaviors.
- However, the values of the characteristics may differ from object to object within a particular class.
Defining a Method

Think about creating methods in terms of three things:

1. **Parameters**
   - Parameters/arguments/values to send to a method?
   - “What does the method need so that it can do its job?”
Defining a Method

Think about creating methods in terms of three things:

1. **Parameters**
   - Parameters/arguments/values to send to a method?
   - “What does the method need so that it can do its job?”

2. **Task**
   - What does the method do?

3. **Return/Results**
   - What does the method produce as a result?
   - What is its final answer?
   - Does it even need to return anything?
Think about creating methods in terms of three things:

1. **Parameters**
   - Parameters/arguments/values to send to a method?
   - “What does the method need so that it can do its job?”

2. **Task**
   - What does the method do?

3. **Return/Results**
   - What does the method produce as a result?
   - What is its final answer?
   - Does it even need to return anything?
The **static** Modifier

- Can be declared for any variable or a method.
- Its value is associated with the class as a whole, not each instance of the class.
The **static** Modifier

- Can be declared for any variable or a method.
- Its value is associated with the class as a whole, not each instance of the class.
- Static variables store ”global” information about the class.
- Static methods are not invoked on a specific instance of a class, instead it is invoked using the name of the class (e.g., `Math` class).
Modularity in Java comes in three varieties: methods, classes and packages:

- **Methods** modularize a program by separating tasks into self-contained units.
Modularity in Java comes in three varieties: methods, classes and packages:

- **Methods** modularize a program by separating tasks into self-contained units.
- **Classes** group together related methods.

Programs are pieced together from user-written methods and classes and pre-written methods and classes from the Java Application Programming Interface, or Java API.
Modularity in Java comes in three varieties: **methods**, **classes** and **packages**:

- **Methods** modularize a program by separating tasks into self-contained units.
- **Classes** group together related methods.
- **Packages** group together related classes.
Modularity in Java comes in three varieties: methods, classes and packages:

- **Methods** modularize a program by separating tasks into self-contained units.
- **Classes** group together related methods.
- **Packages** group together related classes.
- Programs are pieced together from user-written methods and classes and pre-written methods and classes from the Java Application Programming Interface, or Java API.
Java API Packages

- The Java API (Application Programming Interface) contains many separate packages that can be used to make writing complex programs easier.
- Each package focuses on a specific set of tasks and provides pre-written methods:
  - `java.lang` - Fundamentals (Object, String, etc)
  - `java.util` - Utilities (Scanner, Random, etc)
String class

- String str = ‘‘abc’’; is equivalent to:
  String str = new String(‘‘abc’’);
- The + operator joins two strings together.
String class

- String str = ‘‘abc’’; is equivalent to:
  String str = new String(‘‘abc’’);

- The + operator joins two strings together.

- String class is a part of the java.lang package.

- The classes of java.lang package are automatically available for use, no need to import.
The Scanner class in the java.util package is a simple text scanner which can parse primitive types and strings.

We can use the Scanner class to get the input.
The `Scanner` class in the `java.util` package is a simple text scanner which can parse primitive types and strings.

We can use the `Scanner` class to get the input.

We must first create an instance of the `Scanner` as:

```java
Scanner name = new Scanner (System.in); // to read from the terminal,
```
The Scanner class in the java.util package is a simple text scanner which can parse primitive types and strings.

We can use the Scanner class to get the input.

We must first create an instance of the Scanner as:

```java
Scanner name = new Scanner (System.in); // to read from the terminal,
or
Scanner name = new Scanner (File filename); // to read from the file,
```

where

```java
File filename = new File ("input.txt");
```

and `name` is the name you choose for your instance of the Scanner.
Design.
1. Design.
2. Implementation.
Software Development

1. Design.
2. Implementation.
3. Testing.
Software Development

1. Design.
2. Implementation.
3. Testing.
4. Debugging.
Unit Testing a Program with JUnit

Unit testing:
Look for errors in a particular class in isolation.
Unit Testing a Program with JUnit

Unit testing:
Look for errors in a particular class in isolation.

Java library JUnit allows to easily perform unit testing
- For a given class Foo, create another class FooTest to test it, using various ”test case” methods to run.
- Each method looks for particular results and passes / fails.
Unit Testing a Program with JUnit

**Unit testing:**
Look for errors in a particular class in isolation.

Java library **JUnit** allows to easily perform unit testing
- For a given class *Foo*, create another class *FooTest* to test it, using various "test case" methods to run.
- Each method looks for particular results and passes / fails.

JUnit’s "assert" commands:
Put assertion calls in your test methods to check things you expect to be true. If they aren’t true, the test will fail.