Contents

1. Introduction
2. Types and Variables
3. Statements and Control Flow
4. Reading Input
5. Classes and Objects
6. Arrays
7. Methods
8. Scope and Lifetime
9. Utility classes
10. Introduction to Object-Oriented Analysis and Design
Chapter 10: Object-Oriented Analysis and Design

After this chapter you will be able to:

- Describe the properties of object-oriented languages
- Identify inheritance and containment relationships
- Apply fundamental techniques of object-oriented analysis and design
Object-Oriented Language

• An object-oriented (OO) language supports defining objects
• An object is an entity/thing in the application domain
• Can define object data and object operations (methods)
• Objects can have different relationships
  – inheritance
  – containment
Object-Oriented Languages

• Simula (1960s) was the first OO language

• Other OO languages are:
  – Java™
  – C++
  – Eiffel
  – SmallTalk
  – Ada95
Object Definition Example

- Object definitions are called *classes*

```
class Person

Data
Name
Age
Location

Operations
eat()
sleep()
```
Object Instantiation

- Can create multiple *instances* (objects) based on a class

```
class Person
    Name
    Age
    Location

    eat()
    sleep()
```

Name: Jim  Age: 34  Location: home
Name: Alice Age: 29  Location: office
Relationships - Inheritance

- Can define a class by *inheriting* the data attributes and methods of another object
- Inheritance is used to model the Is-A relationship
  
  **class Person**

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>eat()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sleep()</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

  **class Employee**

  | SS#   | work() |

- An Employee Is-A Person
Instantiation with Inheritance

• An Employee object has attributes defined in class Employee and Person

<table>
<thead>
<tr>
<th>Name:</th>
<th>Fred</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age:</td>
<td>40</td>
</tr>
<tr>
<td>Location:</td>
<td>New York</td>
</tr>
<tr>
<td>SS#:</td>
<td>321-11-1232</td>
</tr>
</tbody>
</table>
Other Inheritance Examples

- Chair
  - Desk Chair
  - Lounge Chair
- Phone
  - Rotary Phone
  - Touch Tone Phone
- Adjustable Desk Chair
- Cordless Phone
Relationships - Containment

• Can define a class by *containing* other objects as data attributes

• Containment models the *Has-A* relationship

![Diagram showing containment relationships between Person, Head, and Neck]

• A Person *Has-A* Neck and Head
Other Containment Examples

Person
- Nose
- Heart
- Lungs

Car
- Trunk
- Engine
- Spark plugs
- Starter
OOAD

• Object-oriented analysis and design (OOAD) is the process of describing classes and their relationships

• OOAD is initiated before coding but designs are often modified since software development is an iterative process

• Output of OOAD is a design document that will be referenced by the development team
OOAD Step 1

• List the entities in the application domain

- Console
- Plane
- Passenger cabin
- Missiles
- Tires
- Wing
- Engine
- Airliner
- Exhaust
- Cockpit
- Passenger seat
- Landing gear
- Seat
- Altimeter
- Pilot seat

• What relationships do you see?
OOAD Step 2

- Identify the inheritance (Is-A) relationships

```
Plane  
  └── Airliner
  └── Jetfighter

Seat  
  └── Passenger Seat
  └── Pilot Seat
```
OOAD Step 3

- Identify the containment (Has-A) relationships

Diagram:

- Plane
  - Cockpit
    - Console
    - Pilot seat
  - Engine
    - Exhaust
    - Altimeter
  - Wing
  - Landing gear
    - Tires
OOAD Step 3 continued

• Other Has-A relationships

Airliner
  ↓
Passenger cabin
  ↓
Passenger seat

Jetfighter
  ↓
Missiles
OOAD Step 4

• Draw diagrams that identify the instances of each class

Airliner
- Boeing 727
- DC-10

Jetfighter
- F-15
- Harrier
Further Design

• Can continue the design process by:
  – defining the inputs/outputs of class methods
  – data attributes of each class

• The amount of detail required depends on the size and complexity of the application
References

• Object-oriented Design and Analysis with Applications by Grady Booch
Conclusions/Predictions

Java™ is breakthrough technology that will be important for a long time to come

- Java™ is a versatile language that will be applied in many different ways
- Java™ is more than just a programming language, it integrates:
  - networking
  - multimedia
  - platform independence
It’s Graduation Time

• Congratulations !!