**Chapter 1**

**Self-Review Exercises 1.1**

**Fill in the blanks in each of the following statements:**

1. Computers process data under the control of sets of instructions called **programs**.
2. The key logical units of the computer are the **input unit**, **output unit**, **memory unit**, **central processing unit**, **arithmetic and logic unit**, and **secondary storage unit**.
3. The three types of languages they are **machine languages**, **assembly languages**, and **high-level languages**.
4. The programs that translate high-level language programs into machine language are called **compilers**.
5. **Android** is an operating system for mobile devices based on the Linux kernel and Java.
6. **Release** software is generally feature complete, (supposedly) bug free and ready for use by the community.
7. The Wii Remote, as well as many smartphones, use a(n) **accelerometer** which allows the device to respond to motion.

**1.2 Fill in the blanks in each of the following sentences about the Java environment:**

1. The **java** command from the JDK executes a Java application.
2. The **javac** command from the JDK compiles a Java program.
3. A Java source code file must end with the **.java** file extension.
4. When a Java program is compiled, the file produced by the compiler ends with the **.class** file extension.
5. The file produced by the Java compiler contains **bytecodes** that are executed by the Java Virtual Machine.

**1.3 Fill in the blanks in each of the following statements**

1. Objects enable the design practice of **encapsulation**—although they may know how to communicate with one another across well-defined interfaces, they normally are not allowed to know how other objects are implemented.
2. Java programmers concentrate on creating **classes**, which contain fields and the set of methods that manipulate those fields and provide services to clients.
3. The process of analyzing and designing a system from an object-oriented point of view is called **object-oriented design**.
4. A new class of objects can be created conveniently by **inheritance**—the new class (called the subclass) starts with the characteristics of an existing class (called the superclass), possibly customizing them and adding unique characteristics of its own.
5. **UML (Unified Modeling Language)** is a graphical language that allows people who design software systems to use an industry-standard notation to represent them.
6. The size, shape, color and weight of an object are considered **attributes** of the object’s class.

**1.4 Fill in the blanks in each of the following statements:**

1. The logical unit that receives information from outside the computer for use by the computer is the **input unit**.
2. The process of instructing the computer to solve a problem is called **programming**.
3. **Assembly language** is a type of computer language that uses English-like abbreviations for machine-language instructions.
4. **Output unit** is a logical unit that sends information which has already been processed by the computer to various devices so that it may be used outside the computer.
5. **Memory unit** and **secondary storage unit** are logical units of the computer that retain information.
6. **Arithmetic and logic unit** is a logical unit of the computer that performs calculations.
7. **Arithmetic and logic unit** is a logical unit of the computer that makes logical decisions.
8. **High-level** languages are most convenient to the programmer for writing programs quickly and easily.
9. The only language a computer can directly understand is that computer’s **machine language**.
10. **Central processing unit** is a logical unit of the computer that coordinates the activities of all the other logical units.

**1.5 Fill in the blanks in each of the following statements:**

1. The **Java** programming language is now used to develop large-scale enterprise applications, to enhance the functionality of web servers, to provide applications for consumer devices and for many other purposes.
2. **C** initially became widely known as the development language of the UNIX operating system.
3. The **TCP/IP (Transmission Control Protocol/Internet Protocol)** ensures that messages, consisting of sequentially numbered pieces called bytes, were properly routed from sender to receiver, arrived intact and were assembled in the correct order.
4. The **C++** programming language was developed by Bjarne Stroustrup in the early 1980s at Bell Laboratories.

**1.6 Fill in the blanks in each of the following statements:**

1. Java programs normally go through five phases—**edit, compile, load, verify, and execute**.
2. A(n) **integrated development environment (IDE)** provides many tools that support the software development process, such as editors for writing and editing programs, debuggers for locating logic errors in programs, and many other features.
3. The command java invokes the **Java Virtual Machine (JVM)**, which executes Java programs.
4. A(n) **Java Virtual Machine (JVM)** is a software application that simulates a computer, but hides the underlying operating system and hardware from the programs that interact with it.
5. The **class loader** takes the .class files containing the program’s bytecodes and transfers them to primary memory.
6. The **bytecode verifier** examines bytecodes to ensure that they’re valid.

**1.7 Explain the two compilation phases of Java programs.**

The compilation of Java programs involves two phases:

1. **Compilation to Bytecode:** The Java source code (.java files) is first compiled by the Java compiler (javac) into an intermediate representation called bytecode. Bytecode is platform-independent, meaning it can run on any system that has a Java Virtual Machine (JVM).
2. **Execution by JVM:** The bytecode (.class files) is then executed by the Java Virtual Machine (JVM). The JVM interprets the bytecode and translates it into machine code that the specific computer can understand and execute.

**1.8 One of the world’s most common objects is a wrist watch.**

**Discuss how each of the following terms and concepts applies to the notion of a watch:**

* **Object:** A watch is an object. It is a tangible entity with specific characteristics and behaviors.
* **Attributes:** Attributes of a watch might include its color, size, brand, type of band, and whether it's digital or analog.
* **Behaviors:** Behaviors of a watch include telling time, starting/stopping a stopwatch, setting an alarm, and some watches might have additional behaviors like measuring heart rate.
* **Class:** The class is the general category or blueprint for the object. For example, "watch" is the class, and a specific watch is an instance of that class.
* **Inheritance:** An alarm clock inherits the basic properties of a watch (telling time) but adds a new feature (setting an alarm). So, "alarm clock" can be considered a subclass of "watch".
* **Modeling:** Modeling involves creating a representation of a real-world object or system.

In this case, we are modeling a watch by considering its attributes and behaviors.

* **Messages:** In object-oriented programming, objects communicate by sending messages to each other. For example, a watch might receive a message to "display the time".
* **Encapsulation:** Encapsulation is the bundling of data (attributes) and methods (behaviors) that operate on that data within a single unit or object. The inner workings of how the watch keeps time are hidden, and we only interact with it through its interface (the watch face and buttons).
* **Interface:** The interface is the set of methods or attributes that are accessible to other objects or users. For a watch, the interface includes the watch face, the buttons or crown used to set the time, and any other external ways to interact with the watch.
* **Information Hiding:** Information hiding is the principle of restricting access to certain details of an object. For a watch, the complex mechanisms that make it work are hidden from the user. The user only needs to know how to use the interface (the face and buttons).

**1.9 (Test-Drive: Carbon Footprint Calculator)**

* Some scientists believe that carbon emissions, especially from the burning of fossil fuels, contribute significantly to global warming and that this can be combatted if individuals take steps to limit their use of carbon-based fuels.
* Organizations and individuals are increasingly concerned about their “carbon footprints”.
* Websites such as TerraPass<http://www.terrapass.com/carbon-footprint-calculator/> and Carbon Footprint<http://www.carbonfootprint.com/calculator.aspx> provide carbon-footprint calculators. Test-drive these calculators to determine your carbon footprint.
* Exercises in later chapters will ask you to program your own carbon-footprint calculator.
* To prepare for this, use the web to research the formulas for calculating carbon footprints.

**1.10 (Test-Drive: Body Mass Index Calculator)**

* Obesity causes significant increases in illnesses such as diabetes and heart disease.
* To determine whether a person is overweight or obese, you can use a measure called the body mass index (BMI).
* The United States Department of Health and Human Services provides a BMI calculator at<http://www.nhlbi.nih.gov/guidelines/obesity/BMI/> bmicalc.htm.
* Use it to calculate your own BMI. A forthcoming exercise will ask you to program your own BMI calculator.
* To prepare for this, use the web to research the formulas for calculating BMI.

**1.11 (Attributes of Hybrid Vehicles)**

* Hybrid vehicles are becoming increasingly popular, because they often get much better mileage than purely gasoline-powered vehicles.
* Browse the web and study the features of four or five of today’s popular hybrid cars, then list as many of their hybrid-related attributes as you can.
* Some common attributes include city-miles-per-gallon and highway-miles-per-gallon. Also list the attributes of the batteries (type, weight, etc.).

**1.12 (Gender Neutrality)**

To develop a program that replaces gender-specific words with gender-neutral ones, we can outline a clear algorithm. Here's how the process could work:

**Procedure/Algorithm:**

1. **Input Paragraph:** Accept the paragraph of text to be processed as input.
2. **Load Word List:** Load a pre-defined list of gender-specific words and their corresponding gender-neutral replacements into memory (e.g., a dictionary in Python or a map in Java).
3. **Split Text:** Tokenize (split) the paragraph into individual words, retaining punctuation.
4. **Replace Words:** For each word in the text:
   1. If the word matches a gender-specific term in the list, replace it with the corresponding gender-neutral term.
   2. Retain capitalization and formatting. For example, "Man" becomes "Person" and "MAN" becomes "PERSON."
5. **Reconstruct Text:** Combine the modified words back into a single paragraph with proper spacing and punctuation.
6. **Output Result:** Display the modified paragraph with all gender-specific words replaced.

**Potential Issue:**

Strange terms like "woperchild" may arise if substitutions are performed without proper care. For example:

* If the program replaces "woman" with "person" and then encounters "womanchild" (a compound term), it might transform it into "woperchild" by only replacing "woman" and leaving "child" as is.

**Solution:**

To avoid such issues, the algorithm should:

1. **Check Boundaries:** Ensure replacements occur only for standalone words or well-defined phrases, not as parts of compound words.
2. **Order Replacements:** Process longer words first to prevent partial replacements. For instance, replace "womanchild" entirely before separately replacing "woman."
3. **Refine Tokenization:** Use advanced tokenization methods to distinguish between words and compound terms.

This algorithm provides a structured way to ensure gender-neutral communication while avoiding unintended errors. If you're curious about how this can be translated into Java code, I’d be happy to elaborate!

**Code in java**

import java.util.HashMap;

import java.util.Map;

public class GenderNeutrality {

public static void main(String[] args) {

class GenderNeutralizer {

public static void main(String[] args) {

// Input paragraph

String paragraph = "My son and daughter love their mother and father. A man and a woman should be treated equally.";

// Gender-specific to gender-neutral word mapping

Map<String, String> wordMap = new HashMap<>();

wordMap.put("son", "child");

wordMap.put("daughter", "child");

wordMap.put("mother", "parent");

wordMap.put("father", "parent");

wordMap.put("man", "person");

wordMap.put("woman", "person");

// Process the paragraph

String genderNeutralParagraph = replaceGenderSpecificWords(paragraph, wordMap);

// Output result

System.out.println("Original Paragraph: " + paragraph);

System.out.println("Gender-Neutral Paragraph: " + genderNeutralParagraph);

}

public static String replaceGenderSpecificWords(String text, Map<String, String> wordMap) {

for (Map.Entry<String, String> entry : wordMap.entrySet()) {

String genderSpecific = entry.getKey();

String genderNeutral = entry.getValue();

// Replace all occurrences (case-sensitive)

text = text.replaceAll("\\b" + genderSpecific + "\\b", genderNeutral);

}

return text;

}

}

}

}