Mzumbe University (MU)

CSS 121: Computer Programming with Java

Kilima, Frank Godlove

Programmes: BSc. ITS I, BSc. ICTM I, BSc. ICTB I & BSc. MICT EDU I

Email ID: fgkilima@mzumbe.ac.tz

June 4, 2024

Code of conduct

- Observe the following code of conduct;
  - Be in class on time. Late comers will not be allowed in.
  - Mute or switch off your mobile phones while in class.
  - All communications concerning CSS 121 lectures, tutorials, notes, assignments, tests etc. will be done via CRs.
  - Any excuse for not attending lecture or tutorial sessions should be communicated at the beginning of the lecture/tutorial via CR.
  - Use English for all communications concerning CSS 121.
  - Strictly adhere to the University academic timetable and deadlines.
  - No substitute assignment/test will be given to any students who will fail to write them without good reasons.
  - Read all references provided.
  - Violation of academic integrity will not be tolerated, but dealt with severely in accordance to MU academic regulations.
  - Any communications via emails, including submission of assignments, MUST be done via student's respective MU email (@mustudent.ac.tz) and not otherwise.

Code of Conduct - Cont'd

- Course assessment:
  - Quiz - Many.
  - 2 Assignments @ 10%.
  - 2 tests @ 15%.
  - University Examination (UE) - 50%.
- Marks for assignments, tests or UE can not be compromised or negotiated for.
- Hope to enjoy your maximum cooperation.

Brainstorming

- What is computer program?
- What is computer programming?
- What is computer programming language?
- How do we write computer programs?

Main contents

- Recommended books
- Introduction to computer programming
- Computer programming languages
- Java
- Identifiers
- Reserved words
- comments
- Data types
- Variables
- Java classes
- Method main ()
- Arithmetic operators
- Method println()
- Method
- Programming errors

Recommended References (Books)

1. **Ivor Horton,** *Ivor Horton's Beginning Java, 7e*, John Wiley & Sons inc., 2011.
2. **Herbert Schildt,** *Java The Complete Reference, 9e*, McGraw Hill, 2014.
3. **Herbert Schildt,** *Java A Beginner's Guide* , McGraw Hill, 2018.
4. **Y. Daniel Liang,** *Introduction To Java Programming, 9e*, Pearson Prentice Hall, 2011.
5. **Paul Deitel & Harvey Deitel,** *Java How To Program, 11e*, Pearson, 2017.
6. **Rich Raposa,** *Java in 60 Minutes A Day, 7e*, Wiley Publishing Inc., 2003.

Introduction to computer programming

- Computer program: A special set of stored instructions which control the operations of computer hardware.
- Computer hardware such as CPU, RAM, keyboard etc. are responsible for carrying out computer operations, and therefore need to be instructed.
- Computer programs provide instructions to computer's CPU, which in turn sends signals to respective hardware to execute desired operations.
- Programs give computers ability to carry out all the tasks (functions) they perform.
- Computer program is synonymously known as computer software.

Introduction to computer programming

- A process of developing computer programs.
- It is a multidisciplinary field, which may combine knowledge, skills and experience from other fields including mathematics, arts, computer security, database management, computer networking, linguistics, sciences, AI.
- Diversity in professions involved in the development of any computer program is determined by the purpose of the program being developed.
- **Software development team:** A team of all people involved in developing a certain program (software).

Computer programming language

- It is a formal language consisting of a set of rules (syntax), vocabulary, constructs and utility tools (software) which provide a way of creating computer programs.
- There are hundreds of computer programming languages, each with its own syntax, vocabulary, features, strengths as well as weaknesses.
- Computer programmers: Also known as software developers are computer experts who develop (create) professional computer programs.

Computer programming languages: Types

- Two major types of computer programming languages are:
  - Low level computer programming languages - **READ**
  - High level computer programming languages - **READ**

High level computer programming languages

- They use words and symbols from human languages like English rather than mnemonic codes.
- Computer programs are written using human like languages which are more convenient to use.
- They form the bulk of today's computer programming languages.
- They constantly evolve to incorporate new features in order to improve their security, performance, suitability, usability etc.
- Examples of high level programming languages are BASIC, COBOL, Pascal, FORTRAN, C, C++, Java, Python and Perl.
- High level programming languages allow programmers to use vocabulary of reasonable terms instead of sequence of binary digits.
- They also allow programmers to name storage locations of the computer rather than remembering them.

High level computer programming languages

- Each high level programming language has its own specific and limited set of vocabulary and rules (syntax).
- A set of rules of a programming language is called syntax.
- Learning a programming language means learning a set of vocabulary, syntax and programming logic of that language.
- Programming logic involves writing program's statements in the correct order to produce the desired output (result).
- You may use programming language's syntax correctly but fail to observe the logic.
- Using high level programming language like Java, programmers write series of executable statements, similar to English statements for a program to carry out a specific task.
- Special computer software called **compilers** and **interpreters** are used to translate program's statements from high level programming language into machine code for computer's CPU to execute the instructions specified in statements.

High level computer programming languages

- Compilers and interpreters generate error messages each time they encounter an invalid statement or violation of the language's syntax or vocabulary.
- Errors resulting from invalid programming statements or misuse of the language are called syntax errors.
- Locating and fixing these syntax errors and other types of errors is called debugging.
- Programs written in compiled languages like C++ do execute faster, while those written in interpreted languages like Visual Basic (VB) are easier to develop.
- Java is both a compiled and interpreted language.
- It uses a compiler to translate the program into bytecode, and an interpreter to read the bytecode line by line at runtime (execution).

High level computer programming languages: Strengths

- **High portability:** Largely machine independent and therefore portable. A high level program can be run on different types of platforms (Operating system and CPU) with little or no modifications.
- **Contains fewer code:** Program statements are fewer than their equivalent machine language programs.
- **User friendly:** They look very similar to human (natural) languages like English.
- **Easier to learn and more human understandable:** Because they are user friendly, learning is as easy as learning a human language.
- **Easier to debug and maintain:** It is easier to find errors from programs and remove them.

High level computer programming languages: Weaknesses

- **Less performance:** This is because they need to be compiled (translated) or/and interpreted into appropriate machine language for each CPU platform before they are executed.
- **Need for translators:** Translators (compilers and interpreters) must be installed in order to compile and translate high level programs into their machine language equivalents.

Java: Brief background

- It was built upon the rich legacy inherited from C and C++.
- Invented by James Gosling, Cris Warth, Patrick Naughton, Ed Frank and Mike Sheridon at Sun Microsystems in 1991.
- It was originally known as Oak, but renamed Java and first released in public in 1995.
- Older programming languages like C++ were designed to be **compiled** for a specific computer architecture (CPU).
- Since its first release, Java has continued to evolve significantly.
- The current Java version is Java 21, which is a Long-term support (LTS) released in September 2023.
- Other Java LTS versions are Java 8, Java 11, Java 17.

Motives for inventing Java

- The primary motivation for its invention was the development of platform-independent language for developing software to be embedded in various consumer electronic devices.
- This compelled James Gosling and his colleagues to work on cross-platform computer programming language.
- At the same time, World Wide Web (WWW) was also invented.
- Had the WWW not taken shape at the time Java was invented, Java might have remained a useful but an obscure (unknown) language.
- With the discovery of WWW, Java was propelled to the forefront of computer language design because WWW too demanded portable programs.

Features of Java

- A simple computer programming language whose syntanx is largely based on C and C++ programming languages.
- A multi-platform, class-based, object-oriented programming (OOP), network centric language, general-purpose programming language, fully-featured, and software platform.
- Capable of developing different types of robust applications running on billions of devices including micro-computers, mobile phones, servers, gaming consoles, medical devices etc.
- It is a very stable programming language with;
  - Strong memory allocation
  - Automatic garbage collection mechanism
  - Powerful exception handling
  - Type checking and checks for compile-time errors and run-time errors by compiler and interpreter respectively.
- Contains hundreds of network features to provide massive support for network and distributed based applications.
- Has a built-in ability to support many character sets e.g. ASCII and Unicode.

Features of Java

- Provides comprehensive tools for developing applications with;
  - An interactive Graphical User Interface (GUI)
  - Extensive image processing, graphics programming, access to relational databases and cryptography to name a few.
- It is a fast, highly secure and reliable programming language with many built-in security features such as;
  - Lack of memory pointers
  - Sandboxing
  - Null checking of references
  - Explicit variable declaration
  - Bounds checking of arrays
- Provides a rich set for data structures including lists, queues and arrays.
- It supports multi-threading; It allows a program to have different threads executing independently at the same time.
- It a programming language and software platform available for all major operating systems including Windows, Linux and MacOS, as well as all major hardware architectures.

Features of Java

- Computer platform comprises of hardware type (CPU) and system software (such as OS) of a particular computer.
- It is dynamic: Designed to adapt evolving environment, libraries can add new methods and instance variables without any problem.
- It is free and open source platform - i.e., its software development kit such as JDK can be obtained freely without paying for it, and there exist several free source code for Java stack implementations.
- It is a multi-platform (also known as cross platform, architectural neutral or platform independent) programming language, enabling it to run on a wide range of computer architectures with no or little modifications.
- The platform independence nature of Java enables development of applications which are 'Write Once Run Anywhere (WORA)'
- Platform independence occurs because Java programs do not execute directly on CPU, instead they execute on a runtime environment called **Java Virtual Machine**.

Features of Java

- Java program is first compiled by **Java compiler (javac)** into a bytecode, which is subsequently executed (interpreted) by Java virtual machine (JVM) (also known as **Java interpreter**) for a specific hardware architecture.
- This means Java is both a compiled and interpreted language.
- JVM is part of Java runtime environment (JRE) - a layer of software running on top of a computer's OS to provide libraries and resources required to execute Java programs.
- While Java compiler and JVM are platform dependant, bytecode produced by Java compiler and executed by JVM is platform independent.
- Java bytecode can run on any architecture which supports JVM without need to recompile.
- It is architecturally neutral (platform independent): Allows creation of large scale applications that can run unchanged on different computer platforms.
- The OOP feature allows the development of modular programs and enhances code reusability, code security, clarity etc.

Major applications of Java

- It is widely used for development of;
  - Desktop applications: It can be used to write very powerful standalone or network-based desktop applications
  - Mobile operating systems and mobile applications such as Android operating system and Android mobile applications
  - Big data applications: Used for data science applications and by data processing engines which process complex and massive amounts of real-time datasets
  - Server-side technologies and Internet applications: Used for developing applications running on network servers and many Internet/ web-based applications running on Internet
  - Cloud computing: Because of its 'WORA' nature, it is highly used for development of decentralized cloud-based applications
  - Game applications: For developing various games including mobile, computer and video games
  - Artificial intelligence applications: Used for development of machine learning applications to solve various classification, object detection, natural language processing (NLP) tasks
  - Internet of Things (IoT): Used to program sensors and hardware in edge devices which connect to Internet independently

Java: A compiled and interpreted language

- Source code is a program file made up of program statements written in high level language using a text editor.
- Many high level programming languages' source code files are compiled into object code files compatible with a specific CPU's machine language.
- This is not the case with Java.
- Java compiler (javac) produces an output called bytecode designed to be executed by Java interpreter (Java Virtual Machine (JVM)).
- Bytecode is a binary representation into which a program source code is converted by Java compiler.

Java: A compiled and interpreted language

- Translating Java programs into bytecode makes it easier to run programs in a wide range of environments because only appropriate JVM needs to be implemented for each platform.
- Once appropriate JVM for a given platform is implemented, any Java program can run on it.
- Although details of JVM differ from one CPU platform to another, all JVMs understand the same bytecode.
- If Java programs were compiled to computer's native code, then different versions of the same program would have to exist for each CPU and then fail to be cross-platform.

Java: A compiled and interpreted language

- Because Java programs are isolated from the OS, they are also insulated from particular hardware on which they run.
- This isolation helps to protect Java programs against malicious programs that access computer hardware through OS.
- Java is modelled after C and C++ language, and therefore there is close similarity in the syntax and vocabulary between the two languages.
- Despite this similarity, Java eliminates most of the difficult-to-understand features inherent to C and C++, and with few hardware implementation dependencies.

A compiled language: E.g. C++

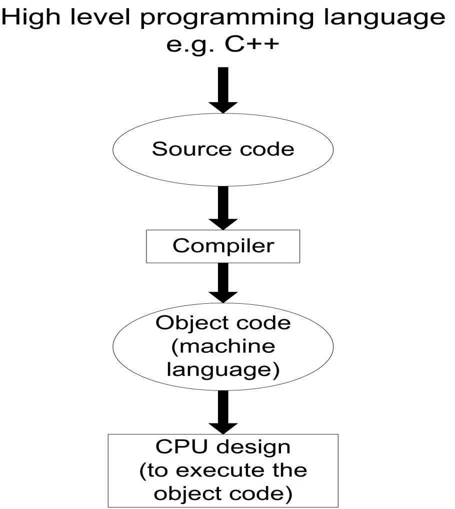


Figure 1: `A compiled High level language like C++`

Java: A compiled and interpreted language

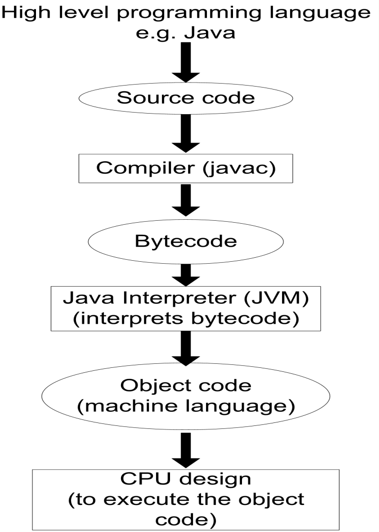


Figure 2: `Compiled and interpreted language like Java`

Java: JRE and JDK

- Java comes shipped in two different ways;
  - Java Runtime Environment (JRE)
  - Java Development Kit(JDK).
- Java Runtime Environment (JRE):
  - Consists of JVM, Java class library and functionalities to start Java programs.
  - You need JRE if you only need to run already compiled Java programs.
  - It does not contain any of the development tools bundled in the JDK.
- Java Development Kit(JDK).
  - In addition to the JRE components, it contains development tools necessary for creating Java programs, and compiler for compiling Java programs.
  - If you need to develop Java programs, you need to install JDK and not JRE.

Major Java Editions

- There exists four major editions of Java;
  - **Java Standard Edition (JSE):** For the development of desktop applications.
  - **Java Enterprise Edition (JEE):** For the development of web applications.
  - **Java Micro Edition (JME):** For development of mobile applications.
  - **Java Cards:** For development of smart card applications.
  - **JavaFX**: Provides a modern, hardware-accelerated graphics and media engine for developing rich desktop applications.
- We will start learning Java Standard Edition (JSE) which enables us to learn Java Core and writing Java desktop applications.
- This will require us to install standard Java Development Kit (JDK).

Requirements for writing Java applications

- A working computer installed with Windows, Linux or Mac OS.
- Well installed and configured Java's Software Development Kit (SDK). I recommend Java Development Kit (JDK).
  - How to download JDK.
  - How to install JDK.
  - How to configure JDK.
- Text editor. I recommend TextPad
- Integrated Development Environment (IDE). I recommend NetBeans (but to be used later)
- Knowledge and skill on Java programming.

Requirements for writing Java applications

- Text editors are good for introducing beginners to programming
- However, they are only suitable for writing short and simple programs.
- IDEs such as Netbeans and Eclipse are appropriate for writing large and complex programs because of the numerous features they offer including;
  - Intelligent code suggestions
  - Code completion
  - Syntax checking
  - Spell checking
  - Framework integration
  - Code replacement
  - Code formatting

Types of Java Programs

- There are three types of Java programs:
  - **Java applets:** Java programs which are embedded into web pages.
  - **Java applications:** These are stand alone Java applications.
  - **Java servlets:** These are Java programs which are executed by a Java enabled web server.
- Java applications are further divided into two categories:
  - **Console applications:** Java applications which support character output on the computer's command prompt, also called command line interface (CLI).
  - **Windowed applications:** Java applications with Graphical User Interface (GUI) components such as menus, drop down boxes, check boxes etc.
- We will start working with console programs and then later with windowed applications.

Creating a first Java program: Steps

- Create a folder, separate from Java installation folder, to store your program source code files.
- Start up your favourite text editor such as TextPad text editor.
- Use text editor (TextPad) to create source (program) code in Java (assuming that you know Java language ☺).
- Save your source code (file) in the created folder with a name that;
  - has the same name as the class name (identifier).
  - not be reserved word.
  - observe other features of identifiers.
  - A .java extension, for instance FirstProgram.java

Creating a first Java program: Steps

- Start up a CMD window (if you are running Windows OS).
- Navigate to the folder in which you have stored your java files.
  - CMD commands: cd, \ and dir.
- Compile a program: javac **program_name**
- Run a program: java **program_name**
- In our case, as the program name is FirstProgram.java, to compile it run the command;

```
javac FirstProgram.java
```

- To execute a compiled program, run the command;

```
java FirstProgram
```

Things to Note

- In compiling Java programs, we run javac command on a Java file having .java extension
- In executing a compiled program, run java on a bytecode file (.class) but WITHOUT .class extension
- For the program to successfully compile, it must not contain any syntax errors. If it does, it will not compile and therefore you will not be able to run it.
- Java is a case sensitive programming language. Upper case letters are not the same as lower case letters. For instance, P is not the same as p.

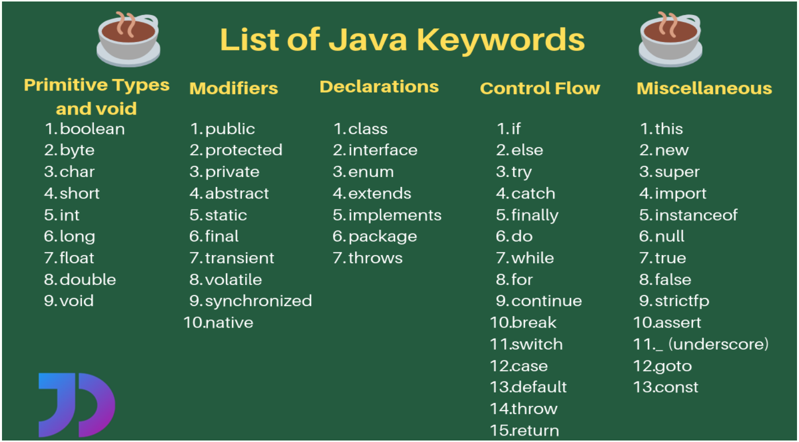
Identifiers

- It is a name given by a programmer to any part of the program such as a class, method, or variable.
- Identifier can have different features depending on the type of the identifier. General characteristics of identifiers are;
  1. It can be of any number of characters (size).
  2. It must start with an alphabetical letter, an underscore (_) or a dollar sign ($).
  3. Cannot start with a numeric digit. Example 6FirstProgram is invalid.
  4. The rest of an identifier can include any character except those used as operators such as +, -, /, *, % or any special character.
  5. Must not include blank space or tabs in the middle of the name. Example First Program is wrong but FirstProgram is valid.
  6. Must not include Java reserved words as they are reserved for the Java language only.

Reserved words

- It is any word that Java uses for special meaning and should not be used by the programmer to name anything (i.e. as an identifier).
- They are used by Java programming language itself for some special purpose
- If used by a programmer as identifier, they will confuse Java, syntax errors will be generated.
- Never use reserved words as identifiers in your program.
- Examples of reserved words are public, class, static, void.
- A comprehensive list of Java keywords is shown in Fig. 3

Reserved words



List of Java Keywords

Primitive Types and void

1. boolean
2. byte
3. char
4. short
5. int
6. long
7. float
8. double
9. void

Modifiers

1. public
2. protected
3. private
4. abstract
5. static
6. final
7. transient
8. volatile
9. synchronized
10. native

Declarations

1. class
2. interface
3. enum
4. extends
5. implements
6. package
7. throws

Control Flow

1. if
2. else
3. try
4. catch
5. finally
6. do
7. while
8. for
9. continue
10. break
11. switch
12. case
13. default
14. throw
15. return

Miscellaneous

1. this
2. new
3. super
4. import
5. instanceof
6. null
7. true
8. false
9. strictfp
10. assert
11. _ (underscore)
12. goto
13. const

Figure 3: `Java reserved words (keywords)`

Comments

- Java compilers tend to ignore (do not execute) all comments contained in the Program.
- Comments is a documentation provided by a programmer within a program used to:
  - Clarify some important points in Java programs.
  - Emphasize something in a program.
  - Change lines of code into comments so that they are not executed.
  - Explain something in the program, such as a function of a program, class, variable etc.
- Two ways of writing comments in Java are using:
  - Double forward slash (//) for each comment line.
  - Putting /* at the beginning of the first comment line and */ at the end of the last comment line.
- Comments are put anywhere in a program.
- Writing useful and good comments in a program is a good programming practice.

Data types

- Data type is a set of values having predefined characteristics.
- For each data type, Java specifies the following;
  - Range of values (minimum to maximum value)
  - Operations to be performed to all values of a particular type
  - Storage mechanism for each data type
- For instance, a byte data type has values ranging from -128 to 127, each byte value takes one byte (8 bits) in memory is manipulated using arithmetic operations (+, -, *, /, %).
- Java is a **statically typed language** i.e., all data storage locations (variables) must be declared before they are used.
- Java is a **strongly typed language** i.e., has its own predefined built-in data types (specifically primitive data types) and does not allow programmers to define their own data types.
- Any data type value is stored in a variable declared to hold a value of that particular type.

Data types

- Assigning a value to a variable that is inconsistent with a declared type generates a compile time error.
- To enhance portability across different architectures (platforms), all data types in Java have a strictly defined range of possible values.
- Compiler allocates memory space for each variable or constant according to its data type.
- An int is a 32-bit data type regardless of the computer architecture allows Java programs to be written and run on any platform without porting on any machine architecture.
- Data types play crucial role in defining the kind of information to be stored or manipulated by program
- Choose the most appropriate data types by considering the range, memory usage and operations to be performed on them.
- Two main data types in Java are:
  - Primitive data types
  - Non-primitive (reference)) data types

Data types

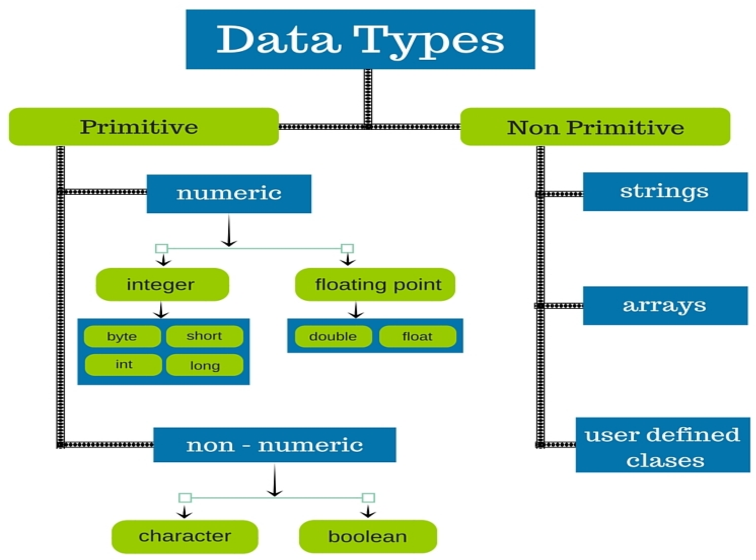


Figure 4: Java's data types

Primitive data types

- Represent single values, rather than complex objects
- They are data types built into Java itself.
- They are referred to as fundamental (basic building blocks) data types in Java.
- All JVM implementations support these primitive data types.
- Although Java is an object oriented programming (OOP) language, primitive data types are not objects.
- Making primitive data types objects would have degraded performance of Java programs significantly.
- There are two distinct categories of primitive data types, which are usually expanded into eight different types.
- Two distinct categories of primitive data types are:
  - Non-numeric primitive data types
  - Numeric primitive data types

Non-numeric primitive data types

- Boolean data type
  - Represents only one of two values; true or false which represent two truth values of logic and Boolean algebra
  - The size of the boolean data type is virtual machine-dependent
  - It is also known as logical data type
  - It has only two values, true and false
  - These values must always be written in lowercase.
- Character data type
  - It is also known as char or single 16-bit Unicode data type
  - Represents symbols in character set, like letters and numbers.
  - Its storage locations (variable) store only one character
  - A character data type has a memory size of 16 bits, which allows characters to be represented as integers.
  - Character data types are unsigned, meaning that they can not store negative value; i.e., can only hold single positive value
  - A character data value must be enclosed with single quotation marks, like 'F'
  - It has a minimum value of '\u0000' (or 0) and a maximum value of '\uffff' (or 65,535 inclusive)

Numeric primitive data types: Integer

- Integer: Integral numeric values such as 78, 100, -11, 1917
- Integer values are signed, i.e., can store both positive and negative values.
- Four integer data types are byte, short, int and long
  - **byte**
    - The smallest integer data type, occupying 8 bits (1 byte) in memory
    - It can be useful for saving memory in large arrays
    - It has a minimum and maximum value of -128 and 127, respectively, i.e, -2^7 to $2^7 - 1$
  - **short**
    - A short value occupies 16 bits (2 bytes) in memory.
    - It has a minimum and maximum value of -32,768 and 32,767, respectively, i.e., -2^{15} to $2^{15} - 1$

Numeric primitive data types: Integer

- **int**
  - It is the most commonly used integer data type in Java.
  - Provides a range larger than one provided by either byte or short.
  - An int value occupies 32 bits (4 bytes) in memory.
  - int values range from -2147483648 to 2147483647, i.e. -2^{31} to $2^{31} - 1$.
  - It covers most of programmers' needs in calculations, control loops, indexing arrays.
- **long**
  - A long value occupies 64 bits (8 bytes).
  - Use this data type when you need a range of values wider than those provided by int data type
  - Values range from -9223372036854775808 to 9223372036854775807, i.e., -2^{63} to $2^{63} - 1$.
  - In order to assign to a variable a data value of type long, it is optionally appended with L (or l).

Numeric primitive data types: Floating point

- Floating point
  - Non-integral numeric values such as 0.123, 345.678, -3.607 etc.
  - Values of this type are called floating numbers.
  - Floating point values are signed
- Two types of floating point data types are;
  - **float**
    - Specified as float values by appending an f (or F) to the value.
    - Float values range from -3.4E38 to +3.4E38 and occupy 32 bits
    - Use a float (instead of double) if you need to save memory in large arrays of floating point numbers.
    - They are represented with approximately 7 decimal digits accuracy.
  - **double**
    - They occupy 64 bits.
    - They range from -1.7E308 to +1.7E308
    - It is a default data type for floating point data values.
    - They are represented with approximately 17 decimal digits accuracy i.e., a number of digits to the right of the decimal point.
    - They are the most commonly used floating point data values.

Non-primitive data type

- It is also called reference data type because it references (stores) a memory location of an instance (object), rather than an object itself.
- It does not store the actual value of the object, but rather store memory address (reference) of an object.
- Some of non-primitive data types are created by the programming language, while others are created by the programmer.
- In Java, non-primitive data types are simply called "objects" because they are created, rather than predefined.
- By default, all non-primitive data types have the same memory size of 8 bytes (i.e, 32 bits).

Non-primitive data type

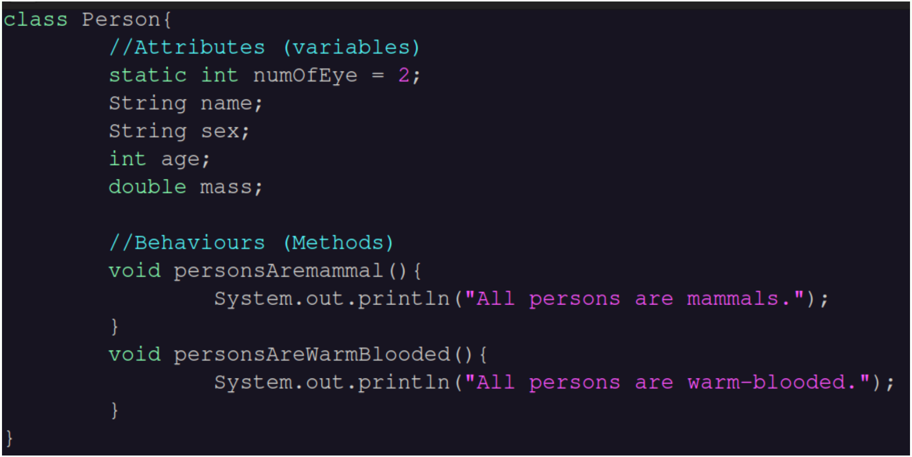
- Examples of non-primitive data types are;
  - Arrays, enumerations
  - Objects of standard (built-in) classes such as String, Random, JFrame, Scanner etc.
  - Objects of user-defined classes (i.e., classes created by programmer)
  - Interfaces
- Everything in Java is object except primitive data types.
- Non-primitive data types are associated with particular classes.
- They represent any instantiable class.
- Non-primitive data types are created by using the **new** keyword.

Primitive vs non-primitive data type

Table 1: **Differences between primitive and non-primitive data types**

Primitive data type	**Non-primitive data type**
They are predefined i.e. they are already defined in Java	They are user-defined data type except String
Can not be used to call (invoke) methods	Used to call (invoke) methods to perform certain operations
Must always have a value	It can be null (a default value)
Starts with lowercase, e.g., int	Starts with uppercase letter String
Memory size depends on data type e.g., byte has 1 byte, int 4 bytes	They all have the same memory size of 8 bytes

Non-primitive data type



```
class Person{
        //Attributes (variables)
        static int numOfEye = 2;
        String name;
        String sex;
        int age;
        double mass;

        //Behaviours (Methods)
        void personsAremammal(){
                System.out.println("All persons are mammals.");
        }
        void personsAreWarmBlooded(){
                System.out.println("All persons are warm-blooded.");
        }
}
```

Figure 5: Definition of the Person class

Non-primitive data type: Creating objects

```
Person aneth = new Person();
```

- Clarifications of the syntax;

  Person:Class name

  aneth:Reference (variable) name to store memory address (reference) of an object - can be any identifier

  =:The assignment operator

  new:keyword used to create objects

  Person( ):Constructor

Variables and variable types

- A variable is named storage location used by the program to store data values.
- They are so called because their contents (values they hold) can change.
- Data values can be numeric, text, boolean, strings or characters.
- A variable must have a specific name (identifier) used to refer to it and a data type describing the category of data value.
- A variable consists of three major parts:
  - **Access specifier (modifier):** public, private, protected or default.
  - **Data type:** Any of the data types discussed, int, float, String, double, char, boolean etc.
  - **Name (identifier):** Any valid identifier. By convention, variable names begin with lowercase first alphabetical letters.

Variable declaration

- A process of creating a named storage location (variable) for storing data value or reference of a certain type.
- It involves specifying a data type and variable name (identifier).
- A variable stores a value of declared type only, and it can not change through out its life time.
- Examples of variable declarations are:

```
int total;
boolean boolValue;
char aLetter;
float floatNum;
```

- Multiple variables of the same data type can be declared in the same statement:

```
int x, y, z;
float fNum1, fNum2, fNum3;
char aletter1, aletter2, aletter3;
```

Variable initialization

- This is the process of fixing initial value to a declared variable.
- Variables can use the initial values in the operations or have them changed.
- Initialization is done using an assignment operator (=).
- The = in Java programming does not have the equality meaning it carries in mathematical calculations.
- Initial values are used as place holders or seat warmers.
- Be careful when assigning initial values to variables, set value which can not cause erroneous results.
- Example of variable initialization is:

```
total = 2;
boolValue = false;
```

- Variables total and boolValue must have been declared before as of type int and boolean, respectively.

Variable declaration and initialization

- Variable declaration and initialization can be done in one statement (step) such as:

```
int total = 2;
boolean boolValue = false;
```

- instead of doing it in two separate statements such as:

```
int total;
total = 2;
```

- This makes programs much shorter.

Types of Variables

- Each variable in Java stores data value of only one type (declared data type).
- Other references refer to variables as fields.
- There are three types of variables:
  - Class variables
  - Instance variables
  - Local variables

Class variables

- Also called static fields because they use the keyword static during their declaration.
- They are associated with the entire class and shared by all class objects.
- They are declared inside the class but outside any method (i.e., not inside any method), commonly after class definition statement
- Only one copy of each class variable exists no matter how many objects are created.
- Exists even if no objects have been created.
- Class variables are commonly used for:
  - Holding constants which are common and required by all class objects such as PI.
  - Tracking data values that are common to all class objects.

Class variables

- They are declared using keyword static with the following syntax:

```
static data_type variableName;
```

- An example of the class variable declaration is:

```
static double bodyWeight;
```

- Declaration and initialization in one statement:

```
static double bodyWeight = 54.5;
```

Class variables holding constants

- Constants do not change throughout program execution.
- They must be protected against both accidental and intentional changes.
- Java provides a way of fixing constants and prevent them from being changed.
- This is done by using keyword final.

```
static final double bodyWeight = 54.5;
```

- The keyword final tells the compiler that the value stored in the variable is a constant and should not be modified anywhere in the program.

Class variables holding constants

- Any variable declared as final must have an initial value assigned in the same statement as it can not be specified later.
- By convention names of class variables holding constants are written in uppercase letters.

```
static final double BODYWEIGHT = 54.5;
```

- This improves readability of the constants within the program.
- Any attempt to change a constant value will generate an error.

Instance variables

- They are also called non-static variables as they do not use keyword static.
- They are declared inside the class but outside any method (i.e., not inside any method), commonly after class definition statement
- They are associated with each class object uniquely.
- Form the bulk of variables declared in program.
- Each instance (object) of the class has its own copy of each instance variable, giving each instance an individuality.
- They are not be operated upon by static methods.
- Referencing an instance variable to a static method generates an syntax error.

Instance variables

Table 2: **Default values for Java's eight primitive data types**

Data type	**Default value**
`byte`	`0`
`short`	`0`
`int`	`0`
`long`	`0L`
`double`	`0.0d`
`float`	`0.0f`
`char`	`'\u0000'`
`boolean`	`false`

Local variables

- Variables declared within the body of the method.
- They exist only within the body of the method.
- They are available only during the execution of the method.
- Their scope (accessibility) is from the point of declaration to the closing brace of that particular method.
- Unlike instance variables, they are not initialized to default values automatically, and therefore they must be initialized to appropriate values before being used.
- Any attempt to use uninitialized local variables generates an error.

Local variables

- Consider the following method:

```
double getSum(double cw, double fe){
     double total = 0.0;  // This is a local variable
     total = cw + fe;
     return total;
}
```

- total is local variable, of type double.
- It must be initialized to any double value before being used.

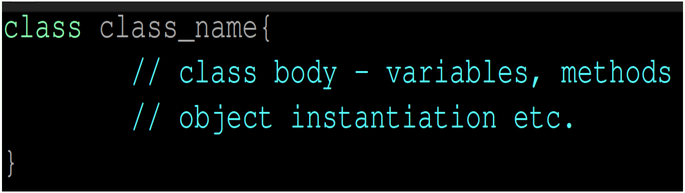
Java classes

- A class, is a set of objects which shares common behaviors and properties (attributes).
- It is a user-defined blueprint or prototype from which objects are created.
- A class defines its own data type, which is an example of reference (non-primitive) data type.
- However simple a Java program is, it must be organized into a functional unit called class.
- Even simple programs that add two numbers (28 + 1), or display messages like "**Hello students, how are you?**" etc. must be organized in a class.
- No any constructs of Java program will work independently without being organized into at least one class.
- Real world programs contain tens, hundreds or thousands of classes.
- A particular instance of the class takes its attributes from the general category.

Java classes

- A class is therefore used to create (instantiate) objects (instances) of its own data type.
- If a program contains one class, then the source code file name must match with this class name.
- If it contains more than one class, then the source code file must match the name of the class that contains the method main().
- As a convention, Java class names should start with an uppercase first letter.
- While it is not a strict rule, it aims at differentiating class names from variable names, and therefore it is a good programming practice to adhere to it.
- To define a class, we use a keyword class, followed by an identifier (class name), followed by braces, also called curly brackets, {}.
- All other constructs (elements) are placed between braces.

Java class - General syntax (form)



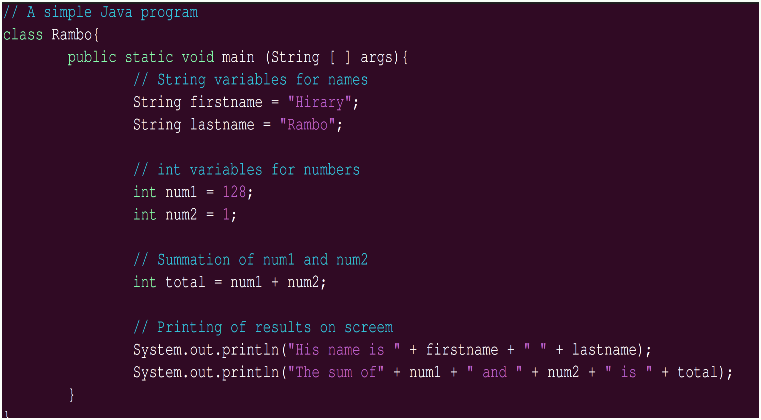
```
class class_name{
        // class body - variables, methods
        // object instantiation etc.
}
```

Figure 6: General syntax of a Java class

Clarification of terms

- **class** keyword
  - It must precede the class_name.
  - It is a reserved word (keyword) that must not be used to name anything else by a programmer.
- **class_name**
  - Any valid identifier.
  - Must start with the uppercase letter: FindAreas, Mzumbe
  - If two or more words are used, capitalize each first letter of each word, e.g. Big4Cat, MzumbeStudents, FindStudents or use an underscore to separate word, such as Find_Students etc.
  - Can be of any length.
  - Must not include blank space or tabs in the middle of the name: e.g Mzumbe Students not a valid name.

Java classes



```java
// A simple Java program
class Rambo{
        public static void main (String [ ] args){
                // String variables for names
                String firstname = "Hirary";
                String lastname = "Rambo";

                // int variables for numbers
                int num1 = 128;
                int num2 = 1;

                // Summation of num1 and num2
                int total = num1 + num2;

                // Printing of results on screem
                System.out.println("His name is " + firstname + " " + lastname);
                System.out.println("The sum of" + num1 + " and " + num2 + " is " + total);
        }
}
```

Figure 7: Simple Java class

Method main ( )

public static void main (String [ ] args)

- The keyword **public**

  Makes a method accessible by any code in the program.
- The keyword **static**

  Ensures that method main() is executed (accessible) even if no objects of the class exist.
- The keyword **void**

  Indicates that main() does not return a value.
- The keyword **main**
- The name (identifier) given to the method
- A pair of parentheses i.e., ()

  It must follow any method name, and it can be empty or carry a parameter/argument.
- **String [ ] args**

  It is an argument of type array which provides a mechanism of accessing data passed to the program.

Arithmetic operators

- Additive operator: +
- Subtraction operator: -
- Multiplication operator: *
- Division operator: /
- Modulus operator: %

Method println()

- The method println() is used to display string of characters (text) on the screen.
- Its general syntax is:

```
System.out.println();
```

- The text to be displayed must be enclosed with double quotes.
- System.out.println(”This is a Java programming class.”);

Method

- A self contained block of code that operates on data values of class.
- Defines operations that can be performed on fields (data values stored in fields).
- It is equivalent to a function in C programming language.
- Major advantage of the method is reusability, can be executed as many times as required.
- Methods help to break a large and complex problem such as calculation into smaller and more manageable chunks.
- A method is executed by calling its name.
- A method may or not return a value when its execution finishes.
- Methods that return a value must finish by executing a return statement.
- An access specifier (attribute) may optionally precede the return type.

Method

- Specifying a return type for a method defines the type for the value that will be returned when the method is executed.
- A returned value must be consistent with the data specified in the method declaration (i.e. return type).
- Methods that do not return any value have a return type specified as void.
- Java method consists of the following major components;
  - Access specifiers: Define access type to the method.
  - The return type: Data type of the value returned by the method or void if it does not return a value.
  - Method name (identifier): A method name that uniquely identifies it in the program.
  - Parameter list:
    - Comma-separated list of input parameters that are defined, preceded by their data type within the enclosed parentheses.
    - If there are no parameters, you must use empty parentheses ()
  - Exception list: List of exceptions you expect the method to throw e.g., ArithmeticException, IOException etc.
  - Method body: A block of code enclosed between braces which execute to perform intended operations.

Method definition

- A method must have a name (valid identifier).
- Conventionally, method name starts with a lowercase letter.
- Parentheses,(), must follow the method name.
- The rest of the method must be enclosed inside braces (curly brackets).
- The general syntax of a method definition is;

```
data_type method_name(){
  // Body of the method
}
```

- Method name is preceded by return type (or void) and optionally by access specifier.
- Return type specifies the data type returned by method which can be any of the primitive data types or reference data type.

Method

- Example of the method definition:

```
int getSum(){
   // Body of the method
}
```

- Body of the method includes the code for operations to be performed by the method.
- For instance;

```
int getSum(){
    sum = num1 + num2;
    return sum;
}
```

- Return type must be consistent with method type, in this case sum must be of type int.
- Variables num1, num2 and sum must have been declared somewhere in the program.

Types of Methods

- There are two types of methods:
  - Instance methods
  - Class methods

Instance methods

- They are executed only in relation to particular objects of class rather than to the class itself.

```
double getSum(){
    double total = 0;
    total = cw + fe;
    return total;
}
```

- They can only be used after instances of the class have been created and can not execute if no object have not been created.
- Variables cw and fe must have been declared somewhere in the program.
- Variable total is a local variable.

Instance methods

- They can access class variables, constants as well as instance variables.
- An instance method can not be called with no reference to any object nor with reference to the class.
- When an instance method is invoked, its name is tacked next to the instance (object) name, separated by a period (.) such as;

```
circle.findCircleArea();
```

Class methods

- They are also called static methods as they are declared using keyword static.
- The static keyword sticks the method to the class rather than to the objects of the class.
- Unlike instance methods, they can be executed even if there are no objects created.
- They can not operate on (reference to) instance variables.
- This is because instance variables may not be created.
- Referencing an instance variable to a static method generates an error.
- They can only access class variables and constants.
- They perform tasks which do not depend on the contents of the objects.
- They apply to the class in which they are defined as whole.

Class methods

- Class methods are only referenced to class variables.

  ```
    static int countObjects(){
       return count;
  }
  ```

- The variable count must have been declared as static variable.
- If not static variable, an error will be generated.
- They are useful for creating and operating on constants and class variables.
- When a class method is invoked, its name is tacked next to the class name, separated by a period,., rather than to any objects of the class.

  ```
        FindArea.countObjects();
  ```

Assignment

1. Write Java program that computes an area of a triangle with;
   - i. At least two instance variables
   - ii. At least one instance method
2. Write Java program called President which display correct full name, age, sex (male or female) and date of assuming office for Tanzanian and Kenyan President.
3. Write Java program with two instance methods, two instance variables and one constant which receives radius (r) and height (h) of the cylindrical body from user through keyboard to compute its surface area $(2\pi rh + 2\pi r^2)$ and volume $(2\pi r^2)$.
4. Write Java program which prompts a user to enter an integer and displays true if an integer is greater than 0 and false otherwise. The program should contain one instance method which return boolean value.
5. Write Java program that prompts user for coefficients of quadratic equation to compute its two x values using general formula assuming that $(b^2 \geq 4ac)$. It should contain;
   - i. At least three instance variables
   - ii. At least two instance methods

Parameterized Method

- Both instance and class methods may have optional one or more values passed to them when they are called.
- Values passed to a method are called **arguments**.
- Variables that receive arguments inside a method are called Parameters.
- For a method to have a value passed, you must first declare a data type and variable name.
- Data type and variable name form a parameter.
- A parameter is declared inside the method's parentheses.

```
int getSum(int a, int b, int c){
   // The rest of the method
}
```

- The return value of method getSum() is of type int.
- int a, int b, int c are the parameters.
- This method must be supplied with 3 values of type int for the three variables when calling it.

Parameterized Method

- An attempt to call this method without supplying the values will generate a compile error.
- It is not allowed to declare parameters of the same type in one statement such as:

```
int getSum(int a, b, c){
   // The rest of the method
}
```

- It is also possible to declare parameters of different data types such as:

```
int getSum(int a, double b, float c){
   // The rest of the method
}
```

- When calling this method, arguments for variables a, b and c must be of the same appropriate data types and provided in the same order as specified in the method.

Parameterized Method

- Arguments must match the parameters in terms of order, number and data type.
- This is called parameter order association.
- If the parameter order association is not observed:
  - A syntax error will be generated.
  - Incorrect result (logic error) will be produced.

Programming error

- A bad code in a computer program that causes it to terminate unexpectedly, produce an incorrect (unexpected) result, or behave in unintended ways.
- Programming errors are commonly referred to as **bugs**.
- **Debugging**: The processing of tracking (identifying) bugs and correcting them.
- Distinguishing them helps programmers to track them more quickly.
- Most common types of programming errors are:
  - Compile-time errors
  - Run-time errors
  - Logical errors

Compile-time errors

- They are also called syntax errors.
- They are kind of errors which occur when the syntax (rules) of programming language has been violated in the program's source code.
- Syntax: Refers to the structure of the program and the rules about the structure.
- Syntax errors are common to new programmers.
- Java compiler can only translate a program that is syntactically correct.
- Program can not run if compilation fails.
- Java compiler is not forgiving, i.e., it does not compile code that contains any errors which violate its syntax.
- If syntax errors are discovered anywhere, compiler prints them (errors) and quits.

Compile-time errors

- Examples of syntax errors include;
  - Omitting a terminator (;) at the end of a statement
  - Unpaired curly or square bracket
  - Mistyping of keywords such as static, void, int, String, public etc.
  - Assigning an incompatible (inconsistent) data to a variable
  - An attempt to use undeclared variable
  - An attempt to use unitialized local variable
  - Not returning or returning a wrong data type to a method that returns a value
  - Returning a value to a void method

Run-time errors

- They are program errors which occur while the program is running, causing it to stop executing and terminates.
- They are so called because they do not appear until you run the program.
- In Java, they occur when interpreter is running the bytecode.
- In Java, run-time errors are called **exceptions**.
- Java is a safe language; meaning that it has few such run-time errors.
- Examples of runtime errors are;
  - Dividing an integer by zero.
  - Trying to open a file that does not exist.
  - Performing an illegal operation that violates security.
  - Accessing an array beyond its size or using a negative index.

Logical errors

- Also called semantics errors.
- They are errors in which the program compiles and runs successfully, but produces unexpected or incorrect outcome (result).
- The program you wrote is not what you wanted to write.
- Identifying logical errors is tricky and may require a lot of time and effort.
- Examples of logical errors are:
  - Using a wrong arithmetic operator, for instance + instead of -.
  - Creating an infinity loop or one with off-by one error.
  - Using a data type such as byte in an operation which produces a value that is outside the range.
  - Calling a different variable but of the same data type in an operation

Test One

1. Define the following terms and give one example for each;
   - (a). Computer programming language
   - (b). Data type
   - (c). Keyword
   - (d). Access specifier
2. Describe any five weaknesses of machine programming languages.
3. Mention key components of any programming language like Java that any programmer needs to learn.
4. Why Java is more portable than other high level programming languages like C+ or C++?
5. List all major categories of primitive data types in Java.
6. List major parts of a Java method.

Individual Drill: ITS I

- Write Java program called WeightOfWater that computes weight of water contained in a rectangular container of dimensions; height, length, and width, representing the container's height, length, and width, respectively. All these dimensions are instance variables whose values are entered by the user through the keyboard. The program should also contain two constants; DENSITY_OF_WATER (1000kg/m3) and GRAVITY (9.8N/kg) representing density of water and gravitational force, respectively. Further, it should contain three instance methods (all returning double values) called computeVolume(), computeMass() and computeWeight() for computing volume, mass and weight of water, respectively.

Individual Drill: ICTB I

- Write Java program called WeightOfWater which computes weight of water contained in a rectangular container of dimensions; height, length and width, representing the container's height, length and width, respectively. All these inputs should be entered by the user through the keyboard and initialized by an appropriate constructor at creation of each object. The program should also contain two constants DENSITY (1000kg/m^3) and GRAV_FORCE (9.81N/kg) for density of water and gravitational force, respectively. Further, it should contain three instance methods (all returning double values) called computeVolume(), computeMass() and computeWeight() for computing volume, mass and weight of water, respectively. The program should finally print the volume, mass and weight of water. Present your work (Fundikira (block B) 207) by Friday, 26^{th} April 2024 at 6.00PM.

Individual Drill: ICTM I

- Write Java program called QuadraticEquation which computes two x values of a quadratic equation expressed as $ax^2+bx+c=0$, using general formula method, assuming that $b^2 \geq 4ac$. The program should contain three instance variables a, b, and c, all of type double, for coefficients a, b and c, respectively. These coefficients must be supplied to the program by the user via the keyboard, and get initialized by an appropriate user-defined constructor at the time of creating an object. The program should also contain two instance methods, both returning double values, called computeFirstX() and computeSecondX() for computing the two x values.

  Deadline and mode of submission: The question be attempted by only 51 students who attended today's tutorial session. The source code file, bearing your full name, programme name and registration number be submitted via email (bearing your name) to fgkilima@mzumbe.ac.tz not later than 10.00AM, 26 April 2024.

Individual Drill 2: ICTM I

- Write a Java program called BankAccount (containing main( )) which allows a user to check balance, deposit cash and withdraw cash from his account by choosing an integer 1, 2 or 3, respectively. If an integer other than 1, 2 and 3 is entered, the program should display a message No such service, and terminate. The program should contain relevant number of instance variables and instance methods. The user should deposit cash not less than 10,000 and withdraw any amount such that the balance left is not less than 10,000. Upon successful completion of any transaction, the program should display the message Thank you for doing business with us, and terminate. The program should also include three well defined packages, balance, deposit and withdraw for CheckBalance, DepositCash and WithdrawCash classes, respectively. No constructor is needed, and the initial deposit amount should be 500,000. Submit the source code by 31^{st} *May*, 2024 at 3.00PM. Assessment will largely base on your understanding of the code and ability to answer questions. Avoid copying of code from anyone.

Part II - Main contents

- Relational operators
- Logical operators
- Order of precedence
- Constructors
- Control structures
- Arrays
- Packages
- Integrated Development Environments (IDEs)Variables
- Basics of Graphical User Interface (GUI)

Relational operators

- Determine the relationship that one operand (value) has to the other, specifically equality and ordering.
- The outcome of the evaluation by relational operators is a boolean value.
- Frequently used in the expressions for most control structures such as if, for and while.
- All primitive data types can be compared using the equality test, ==, and inequality test, !=.
- Only numeric types (integers, floating point and character operands)can be compared using the ordering operators, that is to see which one is greater or less than the other.

Relational operators

Table 3: **Java's relational operators, assume x = 7**

Operator Name	**English Symbol**	**Java operator**	**Example in Java**	**Binary result**
Less than	<	<	5 < x	true
Less than or equal to	≤	<=	5 <= x	true
Greater than	>	>	5 > x	false
Greater than or equal to	≥	>=	5 >= x	false
Equal to	=	==	5 == x	false
Not equal to	≠	!=	5 != x	true

- Placing space between symbols e.g., ! = generates an error.
- All of the above operators are binary operators, that means they operate on two operands at a time.
- Comparing more than two operands will generate a compile time error.

Logical (boolean) operators

- They operate on boolean operands to create a resultant boolean value which is used to determine whether the combined statement (expression) is true or not.
- These operators are;
  - logical NOT
  - logical AND
  - logical OR
  - exlusive OR
- The logical AND, logical OR and XOR operate on binary boolean operands.
- The logical NOT (!), is a unary operator, that it operates on a single boolean operand to invert its state.
- The logical NOT (unary operator) is an operator which operates on a single operand to invert its state.
- Represented using symbols, rather than names and that using names such as NOT instead of ! generates syntax errors.
- Binary logical operators operate from left to right order.

Logical (boolean) operators

Table 4: **Java's logical operators**

Operator	**Symbol**	**Synonym**	**Description**
NOT	!	logical negation	Reverses (inverts) the value of the operand
AND	&	logical AND	Returns true if both operands evaluate to true
AND	&&	conditional AND	Same as &, but if operand on the left is false, it returns false without evaluating operand on the right
OR	|	logical OR	Returns true if at least one operand evaluates to true
OR	||	conditional OR	The same as |, but if operand on the left is true, it returns true without evaluating operand on the right
XOR	∧	logical exclusion	Returns true if one and only one of the operands evaluates to true. It returns false if both operands evaluate to true or both operands evaluate to false

Order of precedence

- Also known as operator precedence.
- Defines the order with which arithmetic operators are executed in a program.
- Multiplication (*), division (/) and modulus (%) are executed before any addition (+) and subtraction (-). The expression;

  ```
  (20 - 3 * 3 - 9 / 3)
  ```

  produces 8.
- You can use parentheses, ( ), in arithmetic calculations to change the order (sequence) of execution.
- For instance, the expression ((20 - 3) * (3 - 9) / 3) produces -34.

Constructors

- **DEMONSTRATION ON HOW** to supply instance variables with values without a constructor.
- When you create an object of a class, a special kind of method called a constructor is always invoked.
- If no constructor is defined for your class, the compiler supplies a default constructor which does nothing.
- The default constructor is also described as the no-arg constructor because it requires no arguments to be specified when it is called.
- The primary purpose of a constructor is to provide programmer with the means of initializing instance variables uniquely for each object created.
- Constructors ensure that instance variables are initialized at the time they are created.
- However, not all instance variables must be initialized with the constructor

Constructors

- Constructor has the following characteristics;
  - It has the same name as the class in which it resides.
  - It is syntactically similar to a method, though it is not a method.
  - It never returns a value and you must not specify a return type not even of type void
  - It is invoked using the new operator when an object is created.
- Constructor does not have a return type because the implicit return type of a class constructor is the class type itself.
- A constructor can have any number of parameter, including none.
- The default constructor has no parameters, as it is indicated by its alternative description the no-arg constructor
- Though it is possible to initialize class variables with constructor, it is rarely done in professional Java programs.

Control structures

- A block of code that dictates the flow of statements in a program.
- The order in which statements in a program are executed is called flow of control.
- Normally statements in a Java program, are executed one after the other in the order in which they are written, a process called sequential execution.
- Java provides constructs which enable programmers to specify the next statement to execute, which is not necessary the next one in the sequence.
- Control structures allow programmers to specify the logic required for statements to execute.

Control structures

- A procedure for solving a problem in terms of actions to execute and the order in which these actions are executed is called an algorithm.
- Determine the actions and the order of execution involved in making ugali.
- Determine the actions and the order of execution involved in taking money from ATM.
- Java provides three types of control structures namely:
  - Sequential control structure
  - Selection control structure
  - Repetition control structure

Sequential control structure

- A default control structure in Java if no any other control structure is specified.
- With sequential control structure, program statements are executed one after another in the order they occur from top to bottom.
- A program can contain any number of statements.
- Consider statements in the main () method, they are usually executed in the order in which they are placed from top to bottom.
- Figure 8 depicts the execution flow of four statements in sequential control structure.

Sequence control structure

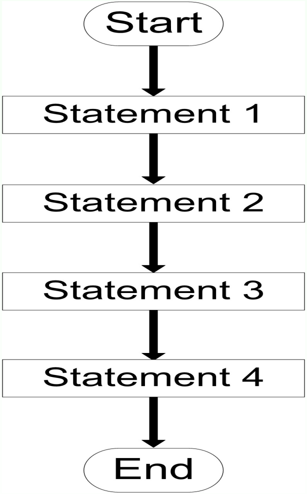


Figure 8: Flow of execution in sequence control structure

Sequence control structure

- In this case, statement 1 will execute first, followed by statement 2, then statement 3 and finally statement 4.
- The statements must be logically placed in a correct order so as to avoid compile time errors or logic errors that may arise if they are wrongly ordered.
- You do not have to add any other code to make sequential execution occur.

Selection control structure

- They are control structures used by programs to choose among alternative courses of action.
- They contain a condition (also called expression) which is evaluated to either true or false.
- If the condition is true, a statement linked to the condition is executed.
- If the condition is false, a statement linked to the condition is ignored (i.e. not executed).
- Java provides three types of selection control structures namely;
  - if
  - if/else
  - if-else-if ladder
  - switch

Selection control structure: if statement

- The if statement is a single selection statement because it selects/performs or ignores a single action (or a single group of action).
- A single group of action (statements) is commonly known as statement block.
- If the condition evaluates to true, statements linked to the if statement are executed, otherwise the statements are ignored (skipped).
- The general syntax of an if statement is;

```
if (expression){
     //statement;
}
```

Selection control structure: if statement

- The expression can be any expression that produces a boolean value.
- For instance testing whether the mark is greater than or equal to 40 and print the message Passed can be written as;

  ```
  if (mark>=40){
       System.out.println(”Passed”);
  }
  ```

- If the value of the expression is true, the statement that follows the if is executed, otherwise it is not executed.
- The indention of the second statement shows that it is subject to the if condition.

Selection control structure: if statement

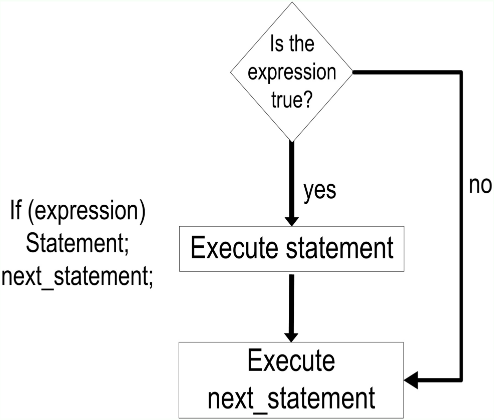


Figure 9: General syntax of the if statement

Selection control structure: if statement

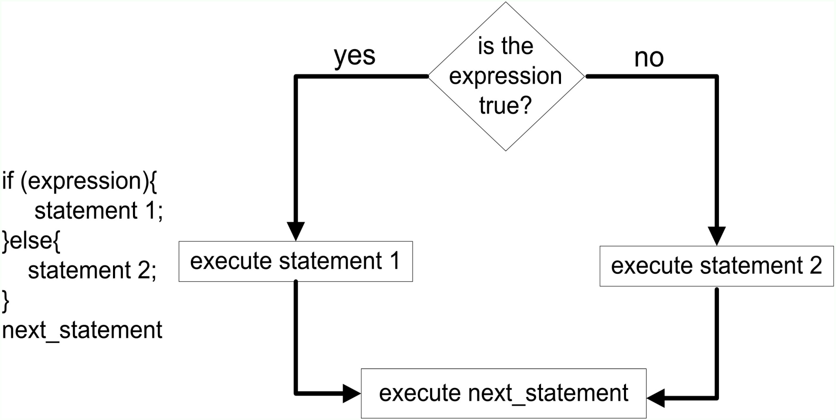
- When you have a statement block in the body of the if statement that is not enclosed with braces, a logic error may be generated.
- In that case, only the statement immediately after the expression will execute if the expression evaluates to true, and not evaluate when the expression evaluates to false.
- The rest will execute whether the expression evaluates to true or not as they will not be considered part of the if body.
- It is always good to enclose the body of the if statement within braces whether you have one statement or statement block.

Selection control structure: if/else

- An if statement can be followed by an optional else clause which executes when the boolean expression (condition) is false.
- The if/else statement provides an explicit choice between two courses of action; one for when the if expression is true and another for when it is false.

```
if (boolean expression){
  // Statements to execute when the expression is true
} else {
  // Statements to execute when expression is false
}
```

Selection control structure: if/else



```
if (expression){
    statement 1;
}else{
    statement 2;
}
next_statement
```

Figure 10: `if/else statement flowchart diagram`

Selection control structure: if-else-if ladder

- It is a variant of the if/else statement.
- Its else statement may contain another if statement,creating a series of if-else statements in which only one if statement execute when its condition is true.
- The conditional expressions are evaluated from top to bottom, and at no time will two or more else if statements execute.
- As soon as the true condition is found, the statement associated with it is executed and the rest of the ladder is bypassed.
- If it erroneously happens that two else-if statements contain the same conditional expression, the first else-if statement in the order will execute.
- If none of the conditions is true, the final (optional) else statement is executed.
- An else clause always associates with the nearest if in the same block that is not already associated with any else.

Selection control structure: if-else-if ladder

```
if(boolean_expression){
  //statement 1;
} else if (boolean_expression){
  //statement 2;
} else if (boolean_expression){
  //statement 3;
} else if (boolean_expression){
  //statement 4;
}else { // This else clause is optional
  //statement 5;
}
```

Selection control structure: switch statement

- Used to select multiple choices (alternatives) based on a set of fixed values for a given expression.
- The expression must produce an integer value other than long; i.e., a value of type byte, short, int and char.
- This means it does not support values of type long.
- Switch statements do not support expressions which produce floating point values, that is double and float values.
- The selections made are determined by the value of the expression placed between parentheses after the keyword switch.
- Possible switch options are defined by one or more case values also called case labels which are defined by using keyword case.
- Case label consists of case keyword followed by a constant value that will select the case, followed by a colon (:).

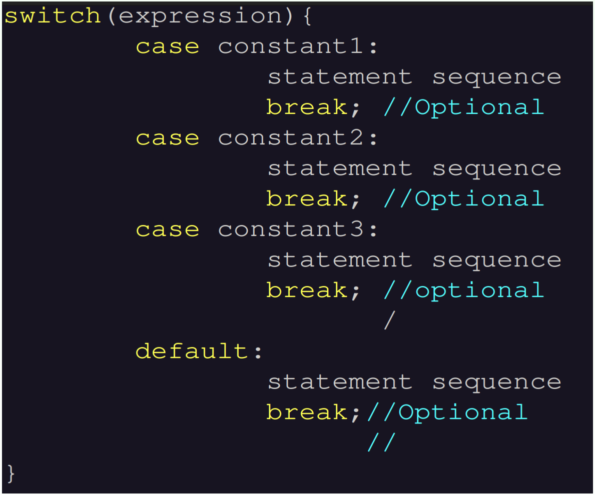
Selection control structure: switch statement

- Statements to be executed when the case is selected follow the case label.
- All case labels and their associated statements are placed between braces for switch statement.
- Switch statement may contain a special case called default, which is selected when the value of the switch expression does not correspond to any of the values for the other cases.
- A value of an expression is successively tested against a list of constant values and when a match is found, the statement sequence associated with that match is executed.
- This means a particular case is selected if its value is the same (matches) with the value of the switch expression.
- Frequently, the expression controlling the switch is simply a variable and the case constant (value) must be literals of a data type compatible with the variable (expression).
- Statements of each case label may be terminated by an optional break statement.

Selection control structure: switch statement

- When encountered within the statement sequence of a case label, the break statement causes the program flow to exit from the entire switch statement and resume at the next statement outside the switch.
- If a break is not placed at the end of the statement sequence associated with a case, all statements following the matching case will be executed until another break or end of the switch is encountered.
- Cases in the switch statement can be written in any order, though it is usually good to sequence them by the order of the case values.
- You can have any number of case statements within the switch.
- The general syntax of the witch statement is;

Selection control structure: switch statement



```
switch(expression){
        case constant1:
                statement sequence
                break; //Optional
        case constant2:
                statement sequence
                break; //Optional
        case constant3:
                statement sequence
                break; //optional
                       /
        default:
                statement sequence
                break;//Optional
                      //
}
```

Figure 11: General syntax of the switch statement

Selection control structure

- if statement: A program that checks whether a mark entered by the user is a pass or not using the set passmark.
- if else statement: Update the above program to include statement when it is not a fail
- if else if: An upgrade of the above program, to also include limiting marks between 0 and 100, inclusive.
- switch statement: A bank program to include methods for depositing cash (depositCash(), withdraw cash (withdrawCash) and checking balance (checkBalance()).
  - Variables deposit, withdraw and deposit (initial and new one)
  - The deposit cash should show

Repetition control structure

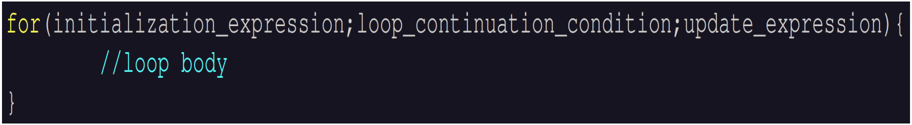
- These are the control structures that enable programs to execute statements repeatedly as long as a condition called the loop-continuation condition remains true.
- They can make a statement or a block of statement to execute in a specified number of times or indefinitely.
- The most common repetition control structures are;
  - for
  - while
  - do while
  - Sentinel controlled while loop
  - while (true)

Repetition control structure: for loop

- The for loop is one of the most commonly used repetition control statements in Java.
- It is declared by using keyword `for`, followed by parentheses containing loop control mechanism, followed by the loop body enclosed in braces.
- The loop control mechanism contains three parts;
  - initialization_expression: Simply called initialization, is an assignment statement that sets the initial value of the loop control variable, such as *int* $i = 0$;
  - loop_continuation condition: It is a boolean expression that determines whether or not the loop will repeat, such as $i < 5$;
  - update_expression: Defines the amount by which the loop control variable will change each time the loop is repeated, such as i++

Repetition control structure: for loop

- The update expression can change by any amount, but usually it changes by one.
- The expression i++ is equivalent to i = i + 1 or i += 1.
- If you want it to change by 2, you can write i += 2.
- The update expression may also involve -, *, / or any valid combination as long as it meets your need.
- If a loop continuation condition evaluates to false from the beginning, the for loop body will not execute
- The general syntax of the for loop is;

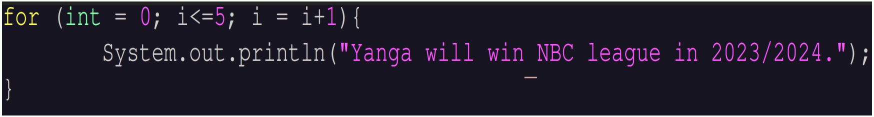


```
for(initialization_expression;loop_continuation_condition;update_expression){
        //loop body
}
```

Figure 12: General syntax of the for loop

Repetition control structure: for loop

- Example of the for loop is;



```
for (int = 0; i<=5; i = i+1){
        System.out.println("Yanga will win NBC league in 2023/2024.");
}
```

Figure 13: for loop

- Quite often, the loop control variable is needed for that purpose and is not used else where in the program.
- When this is the case, it is therefore possible to declare the variable inside the initialization expression of the loop control mechanism and limit its scope within the for loop.
- The for loop can proceed in a positive or negative fashion, though the former is the most commonly used one.
- It is highly recommended to enclose the body of the for loop with braces even if it contains only one statement.

Repetition control structure: while loop

- It contains three major parts;
  - The boolean (conditional) expression
  - Statement (or statement block) to be executed when the condition is true
  - Update expression
- When the boolean expression evaluates to false, the loop terminates and the execution continues with the statement following the loop block.
- The boolean expression is tested at the beginning of the loop, so if it is initially false, the loop body will not be executed at all.
- A statement in the body of the while loop may be a single statement or block of statement.
- The general syntax of the while loop is;

```
while(boolean_expression){
  //statement
  update_operation
}
```

Repetition control structure: while loop

- Example of the while loop is;

```
    while(x < 5){
      System.out.println(”Today is my birthday.....”);
      x++;
    }
```

- Unlike the for loop, the loop counter variable inside the parentheses of the while loop must have been declared and initialized outside the parentheses.
- In the above case, the variable x must have been declared and initialized outside the while loop.
- A compile time error will be generated if you attempt to declare (and initialize) it inside the loop's parentheses.
- Like the for loop, it is highly recommended to enclose the body of the while loop with braces even if it contains only one statement.

Repetition control structure: Sentinel controlled while loop

- It is used where the number of repetitions (iterations)is not known in advance.
- In this case, a condition must be set to enable the loop iterate for any unspecified number of times, and terminate when required.
- It uses a special value called sentinel or signal value which indicates when a loop should terminate.
- Upon entering a sentinel value, the loop will terminate and the next statement in the order of execution after the while loop will execute.
- While a sentinel value can be any number, it must be careful chosen so that it can not be confused with any acceptable values entered by the user.
- The sentinel value may vary from program to program depending on the valid range of the values to be entered by the user.

Repetition control structure: Sentinel controlled while loop

- For instance, a program processing students marks (0 to 100) can have any negative value such as -1 or a positive value greater than 100 as a sentinel value.
- This is because it is not expected a student to score less than 0 or greater than 100 for any test or exam mark.
- The boolean condition uses the most recent mark entered by the user to determine whether the loop should execute again or not.
- Unlike the ordinary while loop, the sentinel controlled while loop does not have an update expression.
- However, great care must be taken to avoid a logical error which may arise if the sentinel value is involved in any computations within the body of while loop.

Repetition control structure: Sentinel controlled while loop

- An example of the sentinel controlled while loop is;

```
while(mark != -1){
  sum += mark;
  System.out.println(”Enter a mark value:  ”);
  mark = input.nextInt();
}
```

- Only when a value entered for mark is not equal to -1, it will be added to sum.
- When -1 is entered, the while loop will terminate and the next statement after the loop will execute.

Repetition control structure: while (true) loop

- The while(true) statement is a better alternative to a sentinel controlled while loop.
- Like all other forms of while loops, there must be a condition for the loop to come out of it, otherwise it will iterate infinitely.
- The most common condition used to terminate the while(true) loop is a break statement.
- The program does not specify the number of times the loop should iterate for the user to enter the marks before the loop terminates.

Repetition control structure: while (true) loop

- Example of the while(true) loop is;

```
while(true){
    System.out.println(”Enter a mark value:  ”);
    mark = input.nextInt();
    if(mark == -1){
      break;
    }
    sum += mark;
}
```

- The if statement executes each time a user enters a value to check if it is equal to -1 or not.
- If it is equal to -1, the break statement executes to terminate the while loop.
- If it is not equal to -1, it is added to sum, and another iteration begins until -1 is entered.

Repetition control structure: do-while loop

- With the do-while loop, the boolean expression (loop continuation condition) controlling the loop is tested at the end of the loop block, that is at the end of each iteration.
- This means that the loop body always executes at least once even if the expression is always false.
- Its general syntax is;

```
do{
  // loop body
  // update_statement
} while(boolean_expression);
```

Repetition control structure: do-while loop

- Since the boolean expression appears at the end of the loop, the statement in the loop is guaranteed to execute at least once before the boolean condition is tested.
- An example of the do-while loop is;

```
do{
    System.out.println(”Enter a mark value:  ”);
    mark = input.nextInt();
    sum += mark;
    i++;
} while(i < 4);
```

Arrays

- **Data structure**: A collection of related data items.
- **Array**: A data structure consisting of related data items of the same data type.
- Other types of data structures include linked lists, stacks, heaps, queues, dictionaries and trees.
- Arrays have the following characteristics;
  - Hold element values of the same data type. No two or more element values can be of different data types.
  - Have fixed size (length) and can not be resized after being created.
  - They are objects, and therefore of reference data type.
  - Elements of arrays can be of either primitive data types or reference data type.

Declaring arrays

- You declare a variable to reference the array and specify the data type of the array elements.

```
        elementDataType arrayRefVar[ ];
```

- The elementDataType can be any data type and all elements in that array must be of that data type.
- The arrayRefVar is the array variable.
- Array (reference) variable names follow the same conventions as other variable names.
- For instance, the following statement creates an array of type int.

```
        int intArray[ ];
```

- Array variable names follow the same conventions as other variable names.
- Arrays can be of any data type, and every element must be of the array's declared type and not otherwise.

Creating arrays

- You can not assign elements to an array unless it has been created.
- To create an array object, you specify data type of the array elements and the number of elements as part of an array creation expression that uses keyword new.
- The general syntax for creating an array is:

  ```
  arrayRefVar = new elementDataType[arraysize];
  ```

- An arraysize specifies the number of elements that can be stored in an array.
- The following statement creates an intArray with the array size of five elements from the declaration made above:

  ```
  intArray = new int[5];
  ```

Declaring and creating an array (combined)

- Declaring an array variable, creating an array and assigning the reference of an array to the variable can be combined in one statement as shown below:
  `elementDataType arrayRefVar [ ] = new elementDataType[arraysize];`
- You can also rewrite the above statement as follows;
  `elementDataType [ ] arrayRefVar = new elementDataType[arraysize];`
- The difference being the position of square brackets, which can be placed before or after the arrayRefVar.

Array initialization

- Array initialization is the process of assigning values to elements of an array.
- After the initialization, element values can be changed any time when necessary.
- Arrays can be initialized with elements at creation or after creation.
  - At creation;

    elementDataType arrayRefVar[ ] = {$value_0$, $value_1$, ..., $value_k$};

    - Observe the absence of new operator and the elementDataType on the right hand side of the assignment operator.
    - Also observe the use of curly brackets in place of square brackets.
    - Initializing an array at creation is appropriate when assigning (initializing) small number of elements.
    - For instance the following statement declares, creates and initialize an array with three elements of type int.

```
int intArray[ ] = {3,6,9};
```

Array initialization

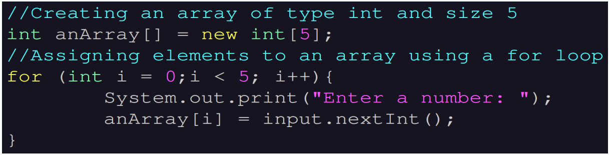
- After creation:
  - This method initializes an array after its creation and is commonly used to initialize one element value at a time.

    ```
    int [ ] intArray = new int[5];
    ```

  - Then assigning values can be done such as;

    ```
    intArray[0] = 5;
    intArray[1] = 12;
    intArray[3] = 45;
    ```

  - Not necessary to initialize all elements at once and in an order as elements can be initialized (or accessed) randomly.
  - Repetition loops such as for and while can also be used to initialize larger arrays much easily.



```
//Creating an array of type int and size 5
int anArray[] = new int[5];
//Assigning elements to an array using a for loop
for (int i = 0;i < 5; i++){
        System.out.print("Enter a number: ");
        anArray[i] = input.nextInt();
}
```

Figure 14: Assigning elements to an array using for loop after creation

Packages

- Package is a uniquely named collection of classes which are related in functionality, scope, or by inheritance.
- To use classes from other packages you must refer to them explicitly by package names or you import them in your source code files.
- Strictly speaking, a package specifies an organized path for folders storing particular classes.
- A package may have a composite name that is a combination of two or more simple names.
- You can specify a package name as any sequence of names separated by periods.

Packages

- It is important to know that package names are case sensitive.
- This means, Scanner is different from scanner
- File path for the package is provided using a period (.) rather than Windows's \ or Linux's and Unix OS's /.

```
package mzumbe.fst.css;
import mzumbe.fst.css;
```

- Classes in packages contain some methods which can be used to perform the operations they have been coded for.
- You can create your own package, define some classes and methods and call them in other classes of your program.
- Two types of packages are;
  - Built-in packages - Java packages included within JDK.
  - User-defined packages - Containing classes written by the user.

Packages

- The use of packages in Java has the following advantages;
  - **Locate and organize classes into units:** This is achieved by grouping together classes that are related in terms of functionality, scope or inheritance.
  - **They reduce name collision:** class names in one package will not interfere with names of other packages' classes of your programs because class names in packages are all qualified by package names.
  - **Provide protection:** This is achieved through the use of Java's access control mechanism that uses access specifiers (access attributes).
  - Making searching/locating and usage of classes, interfaces, enumerations and annotations easier.
  - Packages can be considered as data encapsulation (or data-hiding).

Packages

- You can use packages in Java programs in three different ways.
  - Classes from java.lang such as String and Math are automatically loaded by the JVM and made available to any Java application.
    - ✓ This is because most of classes from this package provide language related support (syntax).
  - Referring to the class by its full name (fully qualified name) which includes its package name.
    - ✓ This is appropriate if the class is used once or twice in the program. For instance the statement;

```
java.util.Scanner input = new java.util.Scanner (System.in);
```

Packages

- You can use packages in Java programs in three different ways.
  - Import individual classes or the whole package using an import statement such as;

    ```
    import java.util.Scanner;
    ```

    - ✓ imports a class Scanner from the package called java.util.
    - ✓ When you want to import all classes from the package, you use the asterisk sign (*). For instance the statement;

      ```
      import java.util.*;
      ```

      imports all classes from the java.util package.
  - ✓ To use the Scanner class, create an object of the class and use any of the available methods found in the Scanner

Creating and using packages

- You need to understand that Java uses a file system directory to store them
- To create your own package, add a package statement as a first non comment line in the source code file of your Java program.
- Only comments and blank lines are allowed to precede the package statement.
- A package statement consists of keyword package followed by package name.
- If you want classes in the package to be accessible outside the package, you must declare the class using keyword public in the first line of the class definition.

Creating and using packages

- Classes definitions not preceded with the keyword public are accessible only from the methods in classes that belong in the same package.
- Finally you must save all the files for classes in package in a directory with the same name.
- A general syntax of package statement is:

  ```
  package packageName;
  ```

- An example of the package declaration is;

  ```
  package mzumbe;
  class FST{
   // class statements eg method definitions
  }
  ```

- A package mzumbe has been created, and the class FST belongs to it.

Creating and using packages

- All classes that are to be included in one package must include the same package statement as the first executable statement and must be saved in the same directory that has the same name as the package.
- You will also have to declare constructors and methods in the classes using keyword public if they are to be accessible outside of the package.
- If you do not declare a class, its constructor, methods or variables public, and you try to access it through import statement in a class of a different package, an access denied error will be generated.

Packages and access specifiers

- They are also called access modifiers or access attributes
- They are elements which control accessibility of class members (data and methods) from other classes in the program
- Access to class members from other classes depends on access specifiers specified for the class members regardless of the package in which the class belongs
- Non-static (instance) members (variables or methods) can not be referenced (accessed) by static (class) members (i.e., instance variables and methods can not be accessed by static (class) variables and methods.
- Static members (variable and methods) can be accessed by both static and non-static members.
- Four access specifiers are public, protected, private and default.
- If no access specifier has been specified for a particular class member, then the member is said to have a default access specifier.
- Default access specifier is essentially absence of an explicit access specifier.

Packages and access specifiers

- public:
  - It is the least restrictive access specifier
  - Public members are accessible to all code (methods) in the program
  - A class declared as public in a package can be accessed from a class within the same package and other packages of the program
  - Classes not declared as public can be accessed only by classes within the same package
- protected:
  - Accessible in any class in the same package and from any subclass(es) within and outside the package in which it is defined.
- private:
  - It is the most restrictive access specifier
  - Private members can only be accessed by code (methods) inside the class
  - Any code that is not a member of the class can not access a private member.
- default:
  - It means no access specifier specified
  - Accessible to any methods in any class in the same package

Packages and access specifiers

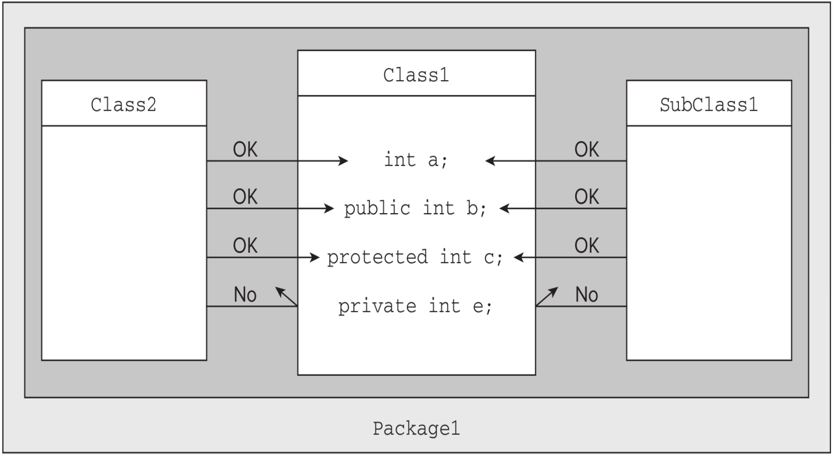


Figure 15: Access specifiers on members within the same package

Object oriented programming: Access specifiers

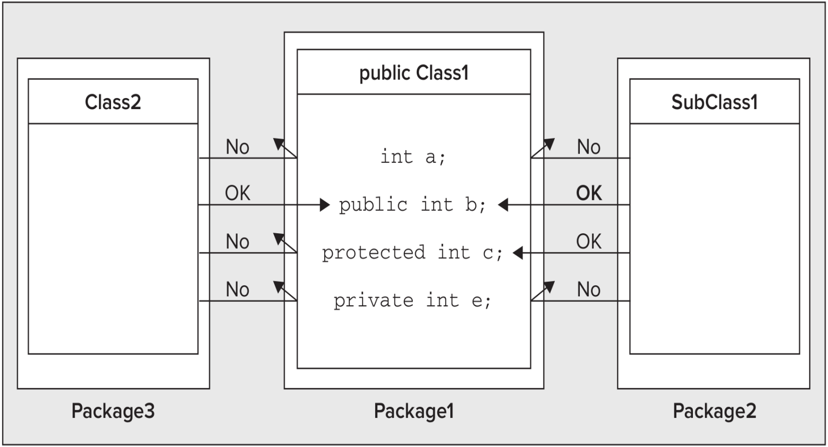


Figure 16: Access specifiers on data members from different packages

Standard packages and classes

- Because of this, Java contains many built-in packages with a huge number of classes, commonly known as standard classes.
- You therefore do not need to write your own classes and methods that perform the same functions performed by any of the standard classes. ”Do not invent your wheel”.
- Standard classes are usually more secure, reliable, uniform throughout the world, reusable, efficient and they reduce effort and time required to write similar classes.

Standard packages and classes

- Commonly used Java's packages include;
  - java.lang: Contains language support classes (e.g classes which defines primitive data types, math operations).
    - ✓ This package is automatically imported into any Java program.
  - java.util: Contains utility classes which implement data structures like linked list, dictionary and support for Date/Time operations.
  - java.io: Contains classes for supporting input/output operations.
  - java.awt:Contains classes for developing Graphical User Interface (GUI) or window-based applications in Java.
  - javax.swing:Contains classes for implementing the components for graphical user interfaces (like button, menus etc.)
  - java.awt.event: Defines classes and interfaces used for event handling in the AWT and Swing.
  - java.net: Contains classes for supporting networking operations.
- Each of these packages contains several classes which in turn contains many methods.

Standard packages and classes

- The Math class which contains most of the mathematical formulas and functions is contained in the java.lang package.
- Examples of Math methods include;

```
Math.max(arg1,arg2):  Returns a max value.
Math.min(arg1, arg2):  Retuns a min value
Math.ceil(arg)
Math.floor(arg)
Math.round(arg)
Math.sqrt(arg):  Finds a square root.
Math.cbrt(arg):  Finds cuberoot of an arg.
Math.pow(x,y):  Find x raised to y.
Math.log10(arg):Finds a base 10 log of an arg.
Math.random()
```

Integrated Development Environment (IDE)

- It is a software suite that consolidates tools essential for software development into a unified platform.
- Provides the programmer with centralized tools for authoring, modifying, testing, debugging, deploying code documentation and others to streamline the development process.
- By integrating multiple tools within cohesive environment, developers can write, test and debug code more efficiently, ultimately contributing to increased productivity.
- Examples of popular IDE for Java programming include NetBeans, Eclipse, Visual Studio, IntellJ
- Among other things, IDE consists of the following;
  - **Source code editor (Text editor):** For creating and editing source code.
  - **Build automation tools:** Performs various activities such as compiling source code, packaging, running tests, deploying software and creating documentation and release notes.
  - **Debugger:** Used to find and remove bugs (errors) in the program source code i.,e. debugging.

Integrated Development Environment (IDE)

- Among other things, IDE consists of the following;
  - **Compiler:** Converts source code into machine readable form for execution by a computer.
  - **Class browser:** Allows developers to navigate the class structure of an object-oriented program.
  - **Object browser:** Allows developers to browse through objects in a project and examine their properties, methods and events.
  - **Class hierarchy diagram:** Allows developers to visualize the structure of the code in an object-oriented programming language.

Integrated Development Environment (IDE)

- Advantages (benefits) of using IDE include;
  - **Syntax highlighting:** Enhances code readability by applyig different colours to different elements such as keywords, variables, classes etc. which helps developers to quickly identify and correct errors.
  - **Intelligent code suggestion:** Suggests possible code that you can type in your program.
  - **Code completion:** Suggests possible code snippets, variable names etc as you type to accelerate coding and reduce the need to type out long and complex expressions and minimize errors.
  - **Snytax checking:** It can identify and minimize coding mistakes and typos.
  - **Intelligent code navigation:** Allows developers to quickly jump between files, classes, packages, methods which is useful in large codebases, enhancing overall productivity.
  - **Spell checking:** Checks spelling of typed code and comments.
  - **Automated code generation:** Automate repetitive coding tasks by generating some codes.

Integrated Development Environment (IDE)

- Advantages (benefits) of using IDE include;
  - **Framework integration:** Provides tools to package applications and integrate them with frameworks.
  - Enforce project, company or language standards:
  - **Breakpoints and debugging console:** Using breakpoints, developers can pause code execution at specific points to facilitate step-by-step debugging.
    - ✓ Debugging consoles provide real-time insight into variable values, function calls and other critical information during debugging process
  - **Variable inspection and tracking:** Provides tools for inspecting variables and tracking their values as code runs.
  - **Code compilation and execution:** Provides real-time compilation, allowing developers to identify syntax errors and compile their code as they type.
  - **Immediate feedback on code changes:** Developers can observe the impact of code changes almost immediately.
  - **Code refactoring tools:** Allows developers to restructure, rename elements such as variables, methods, classes easily and organize imports with minimal efforts, saving time, and ensuring code quality.

Integrated Development Environment (IDE)

- Advantages (benefits) of using IDE include;
  - **Collaborative editing:** Enable multiple developers to work on the same codebase simultaneously which fosters communication and accelerates development process especially in team environment where team members contribute to various parts of code.
  - **Consistent development environment:** Provides consistent development environment across team members through shared project settings, coding standard, version control, unified workspace etc.

Integrated Development Environment (IDE)

- You need to consider the following while selecting an IDE;
  - **Language support:** If the selected IDE supports the programming language you work with.
  - **Platform compatibility:** Confirm that the IDE is compatible with your development platform (Operating system).
  - **Community support:** Look for the IDE that enjoys a strong online presence and a community willing to share knowledge through active forums, discussion groups, and community that can offer help when you face challenges.
  - **Regular updates and bug fixing:** A good IDE should receive regular updates to address bugs, and introducing new features in order to enhance the stability and security of IDE.
  - **Third-party plugin and availability:** Availability of a wide range of third-party plugin or extensions allows developers to customize their IDEs with tools and features that cater their unique development requirements.

Integrated Development Environment (IDE)

- Disadvantages of using IDE include;
  - **Initial setup and configuration:** IDEs, especially those with extensive features may have steeper learning curve during initial setup and configuration.
    - ✓ Setting up of project environment, integrating external tools and preferences which can be time-consuming for new comers.
  - **Mastering advanced features:** IDEs often come with a wide array of advanced features which can be powerful but might require time and effort to master.
  - **Resource intensive:** Some IDEs may demand substantial computational resources which can pose challenges to developers using older hardware or working in resource-constrained environment.
  - **Impact on system performance:** Running IDE alongside other resource intensive applications can impact overall system performance.
    - ✓ IDE processes such as compiling or running intensive analysis may lead to slowdowns especially on less powerful machines.
  - **Feature overload for simple projects:** IDEs may contain comprehensive features which may create a sense of feature overload and make the development environment seem overly complex for the scope of smaller projects.

Basics of Graphical User Interface (GUI)

- All programs we have been creating are called console programs, and they run from the command terminals such as Windows' CMD or Linux command shell.
- While they provided solutions for the problems they were intended to solve, they require special knowledge to use and can rarely be used by users with ordinary computer literacy.
- Many computer users are comfortable to use computer applications which they can easily interact with through graphical user interface such as buttons, menus, texts etc.
- Programs with components such as buttons, text fields are known as windowed applications.
- Java provides an extensive support for GUI, making it one of the best programming languages for developing a wide range of GUI based applications.

Basics of GUI

- Three major frameworks for GUI development in Java are;
  - Abstract Windowing Toolkit (AWT)
  - Swing
  - JavaFX
- AWT and swing reside in the standard packages java.awt and javax.swing, respectively which both contain many classes for writing GUI programs.
- Most components for creating GUI in Java are members (classes and methods) of these packages.
- The java.awt package has been included in Java since the earliest versions of Java.
- AWT was the first framework for GUI development in Java, but due to several limitations, more robust and feature rich frameworks such as swing and JavaFX were developed.
- Limitations of AWT include;
  - Most of its components are dependent on the OS. This means the same component such as a button could have different look and feel depending on the OS the application runs on.
  - Many components of this framework provided limited functionalities.

Basics of GUI

- The next framework was Swing, containing large number of classes which replace or supplement java.awt classes.
- It is more useful and powerful than AWT because;
  - More flexible than AWT as it is implemented purely in Java and does not depend on the OS on which Java runs.
  - Its components contain more and advanced features than AWT.
  - Supports for model-view-controller (MVC) architecture: MVC is an architecture which separates the data model, the user interface, and control logic which simplifies program maintenance.
  - Integration with AWT: Although Swing is an independent frameworks, it integrates very well with AWT.
- Still Swing was found to have a number of issues such as;
  - It is resource-intensive
  - Older look and feel
  - Its performance can be slower than more modern GUI frameworks

Console VS Windowed applications

- Before we start writing GUI-based Java applications, it is important to know the differences between console and windowed applications.
- In console programs, the program code determines the sequence of events.
- Because of this, program statements are ordered depending on the order of events (actions) to be performed by the program.
- In a window-based application, the action(s) performed by the program is driven by the events you do with the GUI.
- Because of this, GUI-based applications are commonly known as event driven programs, because a program performs actions in response to certain events on the application's GUI.
- In event driven program, the part of the program that executes and the subsequent output depend on the action (event) that occurs on the program.

Console VS Windowed applications

- Events performed by user on GUI include;
  - Clicking of the mouse button.
  - Pressing of the window components including button, radio button, check box etc.
  - Entering the text on a text field.
  - Pressing a key on the keyboard.
  - Selecting an item such as text, color etc.
  - Pointing of the window component.
  - Resizing the window etc.
  - Displaying message (text) on the label
- Statements of the GUI programs which execute at any time depend on the event that takes place.
- If a user presses a button or clicks a mouse, then the code that is associated with this event (pressing of a button or clicking of a mouse) will execute.
- For each type of an event, the program could do an event (action) associated with that event.

Basics of GUI

- The code which makes the program produce a certain output when a particular event occurs is called event listener or simply listener.
- Java contains several event listeners for different types of events.
- Most of the event listeners belong to two packages called java.awt.event and javax.swing.event
- These packages contain a large number of classes and methods to respond to various events.
- Listeners in Java are not called in the same way we have been calling methods/statements in console programs.
- Instead they are called automatically when the corresponding event occurs.

Basics of GUI

- Creating GUI-based Java applications will require importing several classes and methods from various packages including;
  - java.awt
  - javax.swing
  - java.awt.event
  - javax.swing.event
- There are many other packages with classes and methods that provide many more functionalities related to GUI.
- This course will make use of Swing framework from javax.swing package.
- However, we still use many classes including Font, Color from java.awt package.

Creating GUI-based Java application - Steps

- To create a GUI based application, be guided with the following;
  - Import the packages for classes and methods you are going to use.
  - Create a window, that is the largest component of the program that holds all other components.
  - Create components of the window.
  - Create the event listener.
- It should be understood that creating components does not automatically make them respond to any events.
- For instance, creating a button, does not automatically make it respond to mouse or key clicks, but you need to create a listener for it.
- Know that most of the classes you will create extend some standard classes.

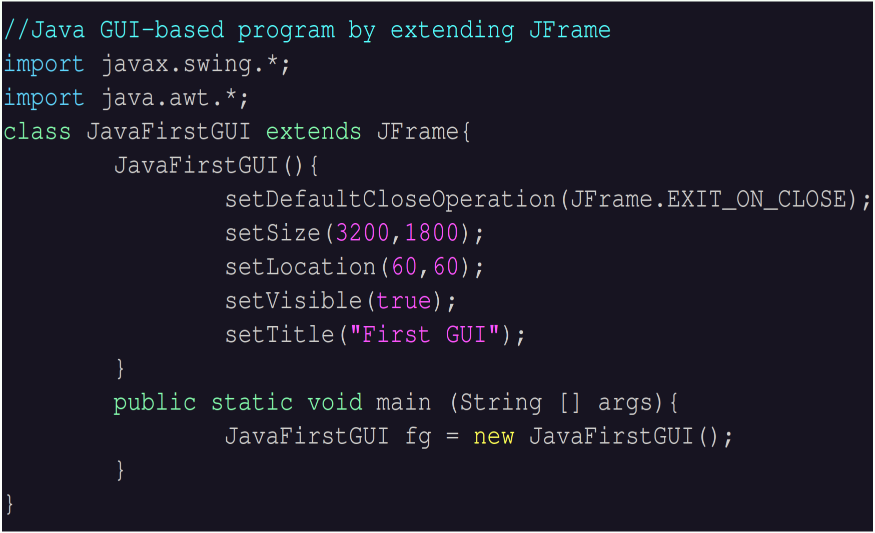
Creating GUI window

- All Java user interfaces (UI) are commonly known as components.
- Many components act as containers to hold other components.
- The largest component in Java application that holds all other components is known as a window, also known as a frame.
- A window (frame) needs to be the first component to create and all other components your application needs to have will be attached (embedded) to it.
- To create a frame, you may use one of the following methods;
  - Defining a class that extends a JFrame class from javax.swing package.
  - Creating a JFrame object to which methods are referenced to
- You then call several standard methods to provide the window with some characteristics.

Creating a window - Extending a JFrame class

- Follow the following steps under this method;
  - Import appropriate classes from awt and swing frameworks.
  - Define a class that extends the JFrame class.
  - Call a constructor with some methods to define some characteristics for the window and/or its components.
- Add more components like panels, textfields, labels, buttons etc.

Creating a window - Extending a JFrame class



```java
//Java GUI-based program by extending JFrame
import javax.swing.*;
import java.awt.*;
class JavaFirstGUI extends JFrame{
        JavaFirstGUI(){
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                setSize(3200,1800);
                setLocation(60,60);
                setVisible(true);
                setTitle("First GUI");
        }
        public static void main (String [] args){
                JavaFirstGUI fg = new JavaFirstGUI();
        }
}
```

Figure 17: The Java Program window: Extending a JFrame class

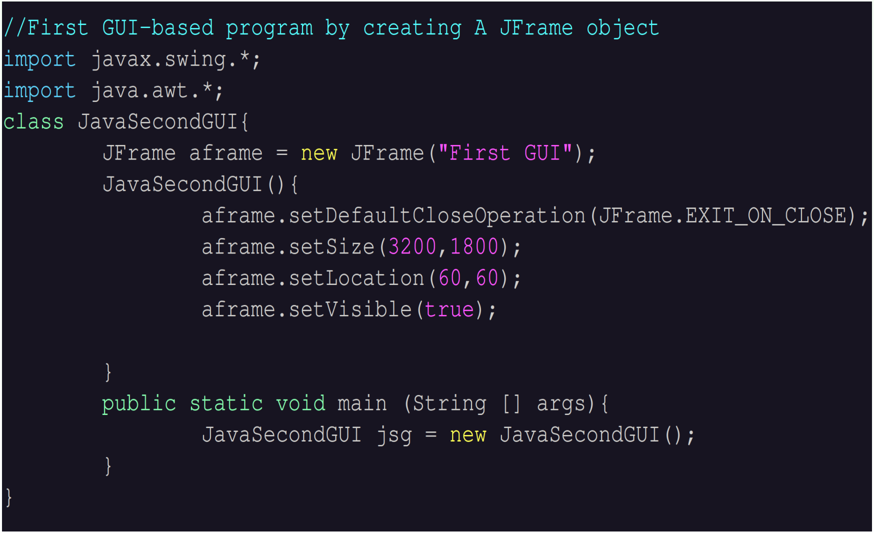
Creating a window - Creating a JFrame class

- Follow the following steps under this method;
  - Create a JFrame object to which all methods defined in the constructor or other methods you will create will be called with reference to
  - This frame object is created like any other object such as;

    ```
    JFrame frameName = new JFrame();
    ```
  - The JFrame constructor may take an argument of type String which will be the title of the window. For instance;

    ```
    JFrame frameName = new JFrame(”FirstGUI”);
    ```
  - Creates a frame object with the title **FirstGUI** Program.
- Look the following two codes which create two very similar windows with the identical characteristics.

Creating a window - Creating a JFrame class



```
//First GUI-based program by creating A JFrame object
import javax.swing.*;
import java.awt.*;
class JavaSecondGUI{
        JFrame aframe = new JFrame("First GUI");
        JavaSecondGUI(){
                aframe.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                aframe.setSize(3200,1800);
                aframe.setLocation(60,60);
                aframe.setVisible(true);

        }
        public static void main (String [] args){
                JavaSecondGUI jsg = new JavaSecondGUI();
        }
}
```

Figure 18: The FirstGUI Program: Creating a JFrame object

Creating a window

- setDefaultCloseOperation(): Controls the behavior of a window when the user tries to close it. It may carry these arguments;
  - JFrame.DO_NOTHING_ON_CLOSE
  - JFrame.HIDE_ON_CLOSE
  - JFrame.DISPOSE_ON_CLOSE
  - JFrame.EXIT_ON_CLOSE
- setVisible(): Sets the size of the window based on the dimensions given. Its arguments are true or false.
  - When it is true the window is displayed, otherwise it is not displayed.
- The setSize() method receives two integer arguments which specify the size of the window in terms of length (left to right) and height (top to bottom).
- The setLocation() method receives two arguments which specify where the top left corner of the program should be placed.
- The setLayout() defines the order (lay out) in which components will be arranged on the window.
  - You need to define a specific layout manager.

Creating a window

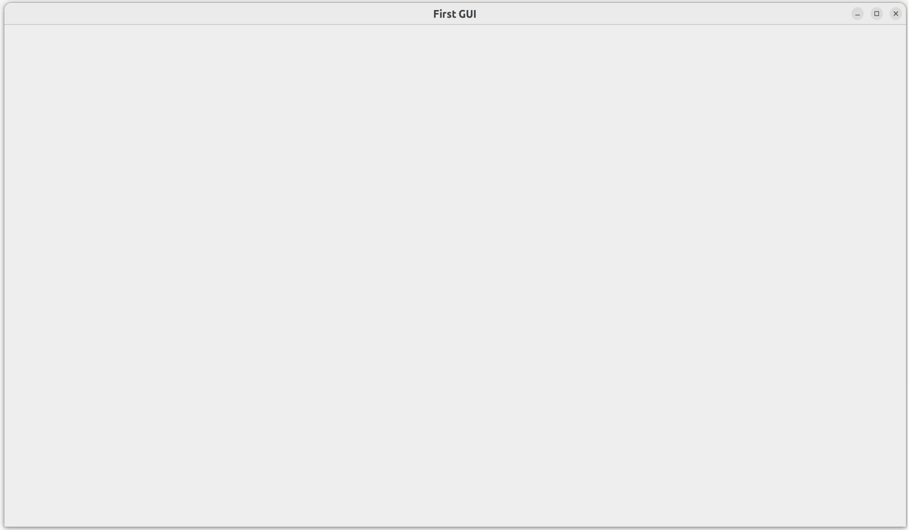


Figure 19: `A window of the first two programs`

Adding smaller components to the window

- The window has two major parts;
  - The menu bar area: Holds the program's menu bar
  - The content pane holds other components
- The above window does not contain any other smaller components such as buttons, label or text field.
- It can only respond to a few events automatically provided by the GUI framework which are resizing the window (maximizing and minimizing) and closing the window.

Adding smaller components to a window

- After the window has been created, you can add other components on it.
- To do this, there are two methods to use, add() and getContentPane() methods.
- When adding any component directly to the window, use the getContentPane() method and not add().
- The add() is used when adding a component to another component, which is then added to the content pane using the getContentPane() method.

Adding smaller components to a window

- A window is GUI's largest component that may hold many other components including;
  - Label
  - Button
  - Panel
  - Radio button
  - Combobox
- Layout managers: For organizing and positioning window components in a suitable manner.
  - FlowLayout manager
  - GridLayout manager
- Graphics tools
  - Colors
  - Fonts

Event handling

- Each component (button, textfield, radio button etc) created must be linked to its respective event listener.
- Event listeners are implemented using interface, not classes.
- There are two ways of creating event handling code in Java which are;
  - Implementing an interface such as ActionListener during class definition.
  - Using an interface.
- We will use the second method - using the interface.
- Each listener interface has the same name as the corresponding event class, but with Listener as a suffix instead of Event.
- Thus, ActionListener interface belongs to the class called ActionEvent, and class FocusEvent, there is an interface called FocusListener.

Event handling

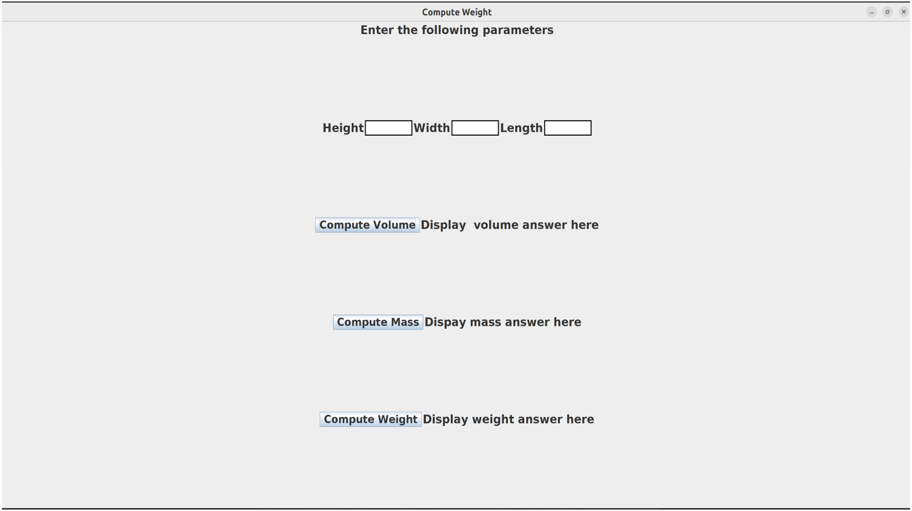
- The methods (actual code to respond to an event) that a listener class must have are defined in the interface.
- Common listener method we will use is actionPerformed( ) that belongs to class ActionEvent.
- It responds to pressing of buttons, changing of text on textfields, selection of choices.
- However, there are other methods for different classes and interfaces which can be used to handle different events.
- A listener method always get an object of type of class to which the interface belongs, for instance, ActionPerformed() must have a parameter of type ActionEvent.
- After implementing the listener method, you must register the listener. This is done by calling the the appropriate method to the component the listener is connected to.
- For button presses we use the method AddActionListener to register the listner.

Customizing behaviors of components

- Java GUI packages contain many classes and methods to change bevaviours of various components in terms of colour, font size, font style, etc.
- The following methods operate on various GUI components;
  - setForeground(): Operates on compoents like labels to change their foreground colours. Eg. label.setForeground(Color.blue)

Group Assignment

- Write GUI-based Java program called WeightOfWaterGUI which computes weight of water contained in a rectangular container of dimensions; height, length and width, representing the container's height, length and width, respectively. All these inputs should be entered by the user through three textfields for height, width and length of the aplication window. The program should also contain two constants DENSITY (1000kg/m^3) and GRAV_FORCE (9.81N/kg) for density of water and gravitational force, respectively. Further, it should contain three instance methods (all returning double values) called computeVolume(), computeMass() and computeWeight() for computing volume, mass and weight of water, respectively. The program should finally print, on the window, the volume, mass and weight of water. All group members must appear in person for the presentation of their program in my office (Fundikira (block B) 207) by Friday, 14th June 2024 at 6.00PM. The program should have a look similar to one shown in Figure 20.



Enter the following parameters

Display volume answer here

Dispay mass answer here

Display weight answer here

Figure 20: `WeightOfWaterGUI window`