[Date]

brunkonjaa

Software Development – Combined Notes (Weeks 1–2)

**1. Programming & Editing Code (Week 1)**

* **Programming** 
  + giving a computer step-by-step instructions.
* **Text Editor:** 
  + basic tool (Notepad, TextPad, Atom). Just writes text.
* **IDE (Integrated Development Environment):** 
  + advanced tool with debugging, compiling, auto-complete, etc.
* **Java Development Kit (JDK):**
  + Needed to compile Java programs.
  + Contains compiler + tools.
  + Must be installed and linked via system PATH.
* **Java Workflow:**
* Write code in .java file.
* Compile → creates .class bytecode file.
* JVM (Java Virtual Machine) executes bytecode on any OS.
* **Java is platform-independent:** 
  + “Write once, run anywhere.”
* **Case sensitive:** 
  + age ≠ Age.
* **Syntax =** 
  + grammar/rules of writing code. Must be followed strictly.
* **Naming Conventions:**
  + Class file names start with **uppercase** (e.g., Example.java).
  + File name must match class name.

**2. Intro to Java**

* **Why Java?**
  + Very popular, stable, widely used in industry.
  + Used in finance, banking, enterprise systems, mobile apps.
* **Platform Independent**:
  + runs on Windows, Mac, Linux, mobile if JVM is installed.
* **WORA**
  + *Write Once, Run Anywhere.*
* **Object-Oriented Programming (OOP)**:
  + **Class** = blueprint/design (e.g., “Human” with attributes like eyes, legs, and actions like walk).
  + **Object** = actual instance of a class (e.g., “Bruno” is a Male object of the Human class).
  + Subclasses can inherit and override parent classes.
  + OOP saves time by reusing designs.
* **Libraries**:
  + collections of reusable code, e.g., Java libraries for graphics, math, networking.

**3. Variables & Data Types**

* **Variable** - named storage location in memory.
* All variables have:
  + **Name** (unique identifier, case-sensitive).
  + **Type** (defines kind of data stored).
* **Java is strongly typed**:
  + must declare variable type upfront.
* **Data Types**:
  + **int**
    - whole numbers (positive/negative, no decimals).
  + **double**
    - decimal/fractional numbers (e.g., 3.14).
  + **String**
    - text (e.g., "Hello World").
  + **char**
    - single character (e.g., 'a', 'Z', '9').
  + **boolean**
    - true or false.
* **Naming rules**:
  + Must begin with a letter.
  + No spaces or special symbols (except \_ or $, though discouraged).
  + Case-sensitive.
  + Use **camelCase** (e.g., userName).
  + Avoid reserved words (e.g., int, class).
* Once declared, a variable’s type cannot change.

**4. Arithmetic Operators**

* **Operators**
  + symbols that perform maths.
    - + addition
    - - subtraction
    - \* multiplication
    - / division
    - % modulus (remainder after division)
* **PEMDAS/BODMAS**
  + Order of operations:
    - Parentheses → Exponents → Multiplication/Division → Addition/Subtraction.
    - Example: 8 / 2 \* (2 + 2) = 16.
* **Example in Java**:
  + - int num1 = 7;
    - int num2 = 8;
    - System.out.println(num1 + num2); // prints 15

**5. JOptionPane (GUI Input & Output)**

* **JOptionPane**
  + part of Swing (used for popup windows).
* **Importing**:
  + import javax.swing.JOptionPane;
* **Output (message box)**:
  + - JOptionPane.showMessageDialog(null, "Hello World!");
      * First parameter (null) = center of screen.
      * Second = message to display.
* **Input (text box)**:
  + - String input = JOptionPane.showInputDialog(null, "Enter your name:");
* **Important**:
  + all input from showInputDialog is treated as **String**.
* **Conversion needed for numbers**:
  + Integer → int num = Integer.parseInt(input);
  + Double → double val = Double.parseDouble(input);

**6. Common 50 Java Errors**

**Compiler Errors (before running)**

1. **“… expected”**
   * Something is missing (usually a semicolon ; or closing bracket )).
   * **Fix**: Check matching pairs of parentheses/braces and missing ;.
2. **“unclosed string literal”**
   * A string wasn’t closed with a quotation mark.
   * **Fix**: End every string with ". Escape quotes inside with \".
3. **“illegal start of an expression”**
   * Code structure is wrong; compiler expected an expression but didn’t find one.
   * **Fix**: Check misplaced braces, keywords, or incomplete statements.
4. **“cannot find symbol”**
   * Refers to variable/class/method that doesn’t exist or isn’t visible.
   * **Fix**: Check spelling, ensure variable is declared and in scope, import classes.
5. **“public class XXX should be in file”**
   * Class name and filename don’t match.
   * **Fix**: Rename file or class so they match exactly (case-sensitive).
6. **“incompatible types”**
   * Assigning one data type to another incorrectly (e.g., int x = "hello";).
   * **Fix**: Convert types (Integer.parseInt, casting) or use correct type.
7. **“invalid method declaration; return type required”**
   * Method declared without specifying return type.
   * **Fix**: Add void, int, etc. Example: public void myMethod().
8. **“method <X> in class <Y> cannot be applied to given types”**
   * Method called with wrong parameters (wrong type or wrong number).
   * **Fix**: Match the method signature (correct number and type of arguments).
9. **“missing return statement”**
   * Method declares a return type but has no return.
   * **Fix**: Add return or change method to void.
10. **“possible loss of precision”**
    * Assigning bigger type to smaller (e.g., double → int).
    * **Fix**: Cast explicitly, or use a larger type.
11. **“reached end of file while parsing”**
    * Missing closing brace } at the end of code.
    * **Fix**: Add missing braces, check indentation.
12. **“unreachable statement”**
    * Code placed after return, break, or continue that will never run.
    * **Fix**: Move or remove unreachable code.
13. **“variable <X> might not have been initialized”**
    * Local variable used before it was assigned a value.
    * **Fix**: Always initialize variables before using them.
14. **“Operator <…> cannot be applied to <X>”**
    * Wrong operator used with incompatible data type (e.g., string1 < string2).
    * **Fix**: Convert types or use valid operators for that type.
15. **“inconvertible types”**
    * Illegal conversion between types (e.g., boolean to int).
    * **Fix**: Avoid incompatible casting.
16. **“missing return value”**
    * Method expects a value but return; is empty.
    * **Fix**: Return correct type.
17. **“cannot return a value from method whose result type is void”**
    * A void method has a return something;.
    * **Fix**: Either change method to return a value or remove return expression.
18. **“non-static variable … cannot be referenced from a static context”**
    * Trying to use an instance variable inside a static method (like main).
    * **Fix**: Make variable static, or create an object instance.
19. **“non-static method … cannot be referenced from a static context”**
    * Similar, but with methods: trying to call an instance method inside main.
    * **Fix**: Call the method from an object instance.
20. **“(array) <X> not initialized”**
    * Array declared but not initialized with size or values.
    * **Fix**: Initialize array with new or values.

**Runtime Exceptions (during execution)**

1. **ArrayIndexOutOfBoundsException**
   * Accessing an index outside valid range of array.
   * **Fix**: Use for (i < array.length), check indices.
2. **StringIndexOutOfBoundsException**
   * Accessing invalid index in a string.
   * **Fix**: Ensure index < str.length().
3. **NullPointerException**
   * Using an object that hasn’t been assigned (null).
   * **Fix**: Initialize object before use, check for null.
4. **NoClassDefFoundError**
   * JVM can’t find compiled class file.
   * **Fix**: Ensure .class file exists in correct location.
5. **NoSuchMethodError**
   * Calling a method that no longer exists.
   * **Fix**: Verify method signature.
6. **NoSuchProviderException**
   * Requested security provider not available.
   * **Fix**: Check Java configuration and installed providers.
7. **AccessControlException**
   * Permission denied for file system/network action.
   * **Fix**: Grant permissions or adjust security manager.
8. **ArrayStoreException**
   * Wrong type stored in array (e.g., putting String in an Integer[]).
   * **Fix**: Use correct array type.
9. **ClassCastException**
   * Invalid object type cast.
   * **Fix**: Use instanceof before casting.
10. **IllegalArgumentException**
    * Method received an illegal or inappropriate argument.
    * **Fix**: Validate input before calling methods.
11. **IllegalMonitorStateException**
    * Using thread synchronization methods incorrectly.
    * **Fix**: Only call wait()/notify() inside synchronized blocks.
12. **IllegalStateException**
    * Object is in inappropriate state for requested operation.
    * **Fix**: Check lifecycle before calling methods.
13. **IllegalThreadStateException**
    * Incorrect use of threads (e.g., starting a thread twice).
    * **Fix**: Only start threads once.
14. **IndexOutOfBoundsException**
    * Generic parent of ArrayIndexOutOfBounds and StringIndexOutOfBounds.
    * **Fix**: Validate indices.
15. **NegativeArraySizeException**
    * Attempting to create an array with negative size.
    * **Fix**: Ensure size >= 0.
16. **NumberFormatException**
    * Invalid number parsing (e.g., Integer.parseInt("abc")).
    * **Fix**: Validate input before parsing.
17. **SecurityException**
    * Security violation, e.g., loading restricted classes.
    * **Fix**: Adjust permissions.
18. **UnsupportedOperationException**
    * Attempting an operation that object doesn’t support.
    * **Fix**: Check collection type or method.
19. **ArithmeticException**
    * Illegal math (e.g., divide by zero).
    * **Fix**: Check divisor before division.
20. **ClassNotFoundException**
    * Attempting to load class not found at runtime.
    * **Fix**: Ensure correct classpath.
21. **CloneNotSupportedException**
    * Object doesn’t implement Cloneable but clone() was called.
    * **Fix**: Implement Cloneable or avoid clone.
22. **EnumConstantNotPresentException**
    * Enum constant doesn’t exist.
    * **Fix**: Verify enum value used.
23. **ExceptionInInitializerError**
    * Error during static initialization block.
    * **Fix**: Debug static code.
24. **FileNotFoundException**
    * File doesn’t exist.
    * **Fix**: Check file path/permissions.
25. **IOException**
    * Generic input/output failure.
    * **Fix**: Handle with try-catch.
26. **InterruptedException**
    * Thread interrupted during wait/sleep.
    * **Fix**: Catch exception and decide how to resume.
27. **MalformedURLException**
    * Invalid URL format.
    * **Fix**: Correct URL string.
28. **SQLException**
    * Database access error.
    * **Fix**: Check DB connection/query.
29. **StackOverflowError**
    * Infinite recursion or too-deep calls exhaust stack.
    * **Fix**: Review recursive methods.
30. **VirtualMachineError (e.g., OutOfMemoryError)**
    * JVM runs out of resources (memory, etc.).
    * **Fix**: Optimize code, increase memory allocation.

**Java Error & Exception Cheat Sheet**

| **Error/Exception Name** | **Plain English Meaning** |
| --- | --- |
| **“… expected”** | **Something missing, usually ; or )** |
| **unclosed string literal** | **String wasn’t closed with "** |
| **illegal start of expression** | **Misplaced or incomplete code structure** |
| **cannot find symbol** | **Variable/class/method not declared, misspelled, or out of scope** |
| **public class XXX should be in file** | **Filename and class name don’t match** |
| **incompatible types** | **Assigning wrong type (e.g., String to int)** |
| **invalid method declaration; return type required** | **Method missing void/int/etc. in declaration** |
| **method … cannot be applied to given types** | **Method call doesn’t match its parameter list** |
| **missing return statement** | **Non-void method doesn’t return a value** |
| **possible loss of precision** | **Assigning larger type into smaller (e.g., double into int)** |
| **reached end of file while parsing** | **Missing } at the end of file** |
| **unreachable statement** | **Code placed after return or break** |
| **variable … might not have been initialized** | **Variable used before it was given a value** |
| **operator … cannot be applied to** | **Wrong operator for type (e.g., < on strings)** |
| **inconvertible types** | **Illegal cast (e.g., boolean → int)** |
| **missing return value** | **Return statement doesn’t provide required type** |
| **cannot return a value from method whose result type is void** | **A void method tries to return a value** |
| **non-static variable … cannot be referenced from a static context** | **Instance variable accessed inside static method** |
| **non-static method … cannot be referenced from a static context** | **Instance method called inside static method** |
| **(array) … not initialized** | **Array declared but not created with new or values** |

Runtime Exceptions

| **Exception Name** | **Plain English Meaning** |
| --- | --- |
| **ArrayIndexOutOfBoundsException** | **Accessing array index outside valid range** |
| **StringIndexOutOfBoundsException** | **Accessing string index outside valid range** |
| **NullPointerException** | **Using object reference that is null** |
| **NoClassDefFoundError** | **JVM can’t find compiled .class file** |
| **NoSuchMethodError** | **Calling a method that doesn’t exist** |
| **NoSuchProviderException** | **Requested security provider not found** |
| **AccessControlException** | **Permission denied for system resource** |
| **ArrayStoreException** | **Wrong type inserted into array** |
| **ClassCastException** | **Invalid object cast** |
| **IllegalArgumentException** | **Invalid method argument** |
| **IllegalMonitorStateException** | **Synchronization methods used incorrectly** |
| **IllegalStateException** | **Method call when object not in correct state** |
| **IllegalThreadStateException** | **Starting thread incorrectly (e.g., twice)** |
| **IndexOutOfBoundsException** | **General parent of array/string out-of-bounds** |
| **NegativeArraySizeException** | **Array created with negative size** |
| **NumberFormatException** | **Invalid number parsing (e.g., “abc” → int)** |
| **SecurityException** | **Security violation (restricted action)** |
| **UnsupportedOperationException** | **Object doesn’t support requested operation** |
| **ArithmeticException** | **Illegal math (e.g., divide by zero)** |
| **ClassNotFoundException** | **Class not found at runtime** |
| **CloneNotSupportedException** | **Object not Cloneable but clone() called** |
| **EnumConstantNotPresentException** | **Enum constant doesn’t exist** |
| **ExceptionInInitializerError** | **Error in static initializer** |
| **FileNotFoundException** | **File missing or path invalid** |
| **IOException** | **General input/output failure** |
| **InterruptedException** | **Thread interrupted during wait/sleep** |
| **MalformedURLException** | **URL format invalid** |
| **SQLException** | **Database access error** |
| **StackOverflowError** | **Infinite recursion/too deep calls** |
| **VirtualMachineError (OutOfMemoryError, etc.)** | **JVM ran out of resources** |