Outline of Test-B (Chapters 1-6)

I. SCOPE

The test aims at testing your understanding of the following topics:

- **Object-Based Concepts**
  Attributes and methods; static vs. non-static; API; field usage; `final` fields; method signature and overloading; method invocation and return; constructors and the role of `new`; memory diagrams; object vs. object reference; etc.
- **Program Development & Java**
  The development cycle; VM, bytecode, and compilation; syntax, runtime and logic errors; statement syntax; declaration; I/O via the `type.lang` package; primitive types; expression evaluation; automatic vs. manual casts; assignment, relational, and boolean operators; selection; iteration; etc.
- **Using a standard or a given API**
  Accessing fields and invoking methods; static versus not-static; declaring local variables and constants; carrying out assignments; I/O; input validation; output formatting; appropriate usage of selection and looping constructs; operators; etc.

II. FORMAT

The test achieves its objectives through two groups of questions of weights ~40% (for A) and ~60% (for B):

A. Multiple-choice or tracing questions in which given a Java program or a fragment thereof, you are asked to identify syntax / logic errors, describe what the fragment is doing (its functionality), and/or state the fragment's output(s) given its input(s).

B. You will be asked to write an app (or a fragment thereof) that accomplishes a stated task. If the task involves a new class, its API will be given. Otherwise you rely on your knowledge of the API of the `String` class, the `Math` class, or the classes in `type.lang`.

Note:
You are assumed to have memorized the names of the primitive types; the arithmetic, relational, and boolean operators; the assignment algorithm; and the main features of the API of the classes: `type.lang.IO`, `type.lang.SE`, `java.lang.Math`, and `java.lang.String` Nevertheless, the following sheet will be provided:

Data Sheet for Test-B

String Methods (invoke on a string s)	`char` **`charAt(int p)`** Returns the character at position# p in s.
`boolean` **`equals(String t)`** Returns *true* if s and t have equal contents.	`int` **`compareTo(String t)`** Returns a negative number if s<t, zero if s=t, and a positive number if s>t.

`int` **`indexOf(String t, int f)`**
Looks for the string t within s, starting at position# f in s. Returns the position in s where the match was found. Returns -1 if no match was found.
	`Integer.`**`parseInt`**`(s)`
	`Double.`**`parseDouble`**`(s)`
	Static methods to convert a string **s** that contains a number to a primitive type.

`int` **`indexOf(String t)`** Looks for the string t within s (as above), starting at the beginning of s.	`String` **`trim()`** Returns the same content as s but with any leading/trailing white-space removed.
`String` **`substring(int f,int t)`** Returns all characters in s with position numbers ≥ f and < t.	**Static methods in Math**
`String` **`substring(int f)`** Returns a substring of s that begins at f and extends to the end of s.	`double` **`abs(double x)`** Returns the absolute value of x.
`String` **`replace(char x,char y)`** Returns a string with all occurrences of character x in s replaced by y.	`double` **`pow(double x, double y)`** Returns x raised to y.
`String` **`toUpper/LowerCase()`** Returns a string of all characters in s converted to upper / lower case.	`double` **`rint`**`(double a)` Returns the closest `double` value to `a` that is equal to a mathematical integer.