Aims (Skills)

To devise algorithmic solutions to problems and to be able to code, validate, document, execute and debug the solution using the Java programming system.

There will be two papers in the subject:

***Paper I**: Theory…….. 3 hours…70 marks*

***Paper II**: Practical……. 3 hours…30 marks*

SECTION A

Basic Computer Hardware and Software

1.Numbers

Representation of numbers in different bases and interconversion between them (e.g. binary, octal, decimal, hexadecimal). Addition and subtraction operations for numbers in different bases.

Introduce the positional system of representing numbers and the concept of a base. Discuss the conversion of representations between different bases using English or pseudo code. These algorithms are also good examples for defining different functions in a class modelling numbers (when programming is discussed). For addition and subtraction (1's complement and 2's complement) use the analogy with decimal numbers, emphasize how carry works (this will be useful later when binary adders are discussed).

2.Encodings

(a) Binary encodings for integers and real number using a finite number of bits (sign magnitude, 2's complement, mantissa exponent notation).

Signed, unsigned numbers, least and most significant bits. Sign-magnitude representation and its shortcomings (two representations for 0, addition requires extra step): two's-complement representation. Operations (arithmetic, logical, shift), discuss the basic algorithms used for the arithmetic operations. Floating point representation: normalized scientific notation, mantissa and exponent). Single and double precision.

(b) Characters and their encodings (e.g. ASCII, ISCII, Unicode).

Discuss the limitations of the ASCII code in representing characters of other languages. Discuss the Unicode representation for the local language. Java uses Unicode, so strings in the local language can be used (they can be displayed if fonts are available)- a simple table lookup for local language equivalents for Latin (i.e. English) character strings may be done. More details on Unicode are available at www.unicode.org.

3. Propositional logic, Hardware implementation, Arithmetic operations

a) Propositional logic, well-formed formulae, truth values and interpretation of well-formed formulae, truth tables.
Propositional variables; the common logical connectives ((not)(negation), ∧ (and)(conjunction), ∨ (or)(disjunction), ⇒ (implication), ⇔ (equivalence)); definition of a

*well-formed formula (wff); representation of simple word problems as wff (this can be used for motivation); the values **true** and **false**; interpretation of a wff; truth tables; satisfiable, unsatisfiable and valid formulae.*

b) Logic and hardware, basic gates (AND, NOT, OR) and their universality, other gates (NAND, NOR, XOR, XNOR), half adder, full adder.

Show how the logic in (a) above can be realized in hardware in the form of gates. These gates can then be combined to implement the basic operations for arithmetic. Tie up with the arithmetic operations on integers discussed earlier in 2 (a).

SECTION B

The programming element in the syllabus is aimed at algorithmic problem solving and **not** merely rote learning of Java syntax. The Java version used should be 5.0 or later. For programming, the students can use any text editor and the javac and java programs or any other development environment: for example, BlueJ, Eclipse, NetBeans etc. BlueJ is strongly recommended for its simplicity, ease of use and because it is very well suited for an 'objects first' approach.

4. Introduction to Object Oriented Programming using Java

Note that topics 5 to 12 should be introduced almost simultaneously along with Classes and their definitions.

5. Objects

a) Objects as data (attributes) + behaviour (methods or methods); object as an instance of a class.

Difference between object and class should be made very clear. BlueJ (www.bluej.org) and Greenfoot (www.greenfoot.org) can be used for this purpose.

b) Analysis of some real-world programming examples in terms of objects and classes.

Use simple examples like a calculator, date, number etc. to illustrate how they can be treated as objects that behave in certain well-defined ways and how the interface provides a way to access behaviour. Illustrate behaviour changes by adding new methods, deleting old methods or modifying existing methods.

c) Basic concept of a virtual machine; Java Virtual Machine (JVM); compilation and execution of Java programs (the javac and java programs).

The JVM is a machine but built as a program and not through hardware. Therefore it is called a virtual machine. To run, JVM machine language programs require an interpreter. The advantage is that such JVM machine language programs (.class files) are portable and can run on any machine that has the java program.

d) Compile time and run time errors; basic concept of an exception, the Exception class, try-catch, throw, throws and finally.

Differentiate between compile time and run time errors. Run time errors crash the program. Recovery is possible by the use of exceptions. Explain how an exception object is created

and passed up until a matching catch is found. This behaviour is different from the one where a value is returned by a deeply nested method call.

6. Primitive values, Wrapper classes, Types and casting

Primitive values and types: byte, int, short, long, float, double, boolean, char. Corresponding wrapper classes for each primitive type. Class as type of the object. Class as mechanism for user defined types. Changing types through user defined casting and automatic type coercion for some primitive types.

Ideally, everything should be a class; primitive types are defined for efficiency reasons; each primitive type has a corresponding wrapper class. Classes as user defined types. In some cases types are changed by automatic coercion or casting – e.g. mixed type expressions. However, casting in general is not a good idea and should be avoided, if possible.

7. Variables, Expressions

Variables as names for values; named constants (final), expressions (arithmetic and logical) and their evaluation (operators, associativity, precedence). Assignment operation; difference between left-hand side and right-hand side of assignment.

Variables denote values; variables are already defined as attributes in classes; variables have types that constrain the values it can denote. Difference between variables denoting primitive values and object values – variables denoting objects are references to those objects. The assignment operator = is special. The variable on the LHS of = denotes the memory location while the same variable on the RHS denotes the contents of the location e.g. i=i+2.

NOTE: Library functions for solving expressions may be used as and when required.

8. Statements, Scope

Statements; conditional (if, if else, if else if, switch case) ternary operator, looping (for, while, do while), continue, break; grouping statements in blocks, scope and visibility of variables.

Describe the semantics of the conditional and looping statements in detail. Evaluation of the condition in conditional statements.

Nesting of blocks. Variables with block scope, method scope, class scope. Visibility rules when variables with the same name are defined in different scopes.

9. Methods and Constructors

Methods and Constructors (as abstractions for complex user defined operations on objects), methods as mechanisms for side effects; formal arguments and actual arguments in methods; Static methods and variables. The ***this*** operator. Examples of algorithmic problem solving using methods (number problems, finding roots of algebraic equations etc.).

Methods are like complex operations where the object is implicitly the first argument. Operator this denotes the current object. Methods typically return values (changes made inside methods persist after the call for object values). Static definitions as class variables

and class methods visible and shared by all instances. Need for static methods and variables. Introduce the main method – needed to begin execution. Constructor as a special kind of method; the new operator; multiple constructors with different argument structures; constructor returns a reference to the object.

10. Arrays, Strings

Structured data types – arrays (single and multidimensional), strings. Example algorithms that use structured data types (searching, finding maximum/minimum, sorting techniques, solving systems of linear equations, substring, concatenation, length, access to char in string, etc.).

Storing many data elements of the same type requires structured data types – like arrays. Access in arrays is constant time and does not depend on the number of elements. Sorting techniques (bubble, selection, insertion), Structured data types can be defined by classes – String. Introduce the Java library String class and the basic operations on strings (accessing individual characters, various substring operations, concatenation, replacement, index of operations).

SECTION C

11. Basic input/output using Scanner and Printer classes.

a) Basic input/output using Scanner and Printer classes.

Input/output exceptions. Tokens in an input stream, concept of whitespace, extracting tokens from an input stream (String Tokenizer class). The Scanner class can be used for input of various types of data (e.g. int, float, char etc.) from the standard input stream. Only basic input and output using these classes should be covered.

Discuss the concept of a token (a delimited continuous stream of characters that is meaningful in the application program – e.g. words in a sentence where the delimiter is the blank character). This naturally leads to the idea of delimiters and in particular whitespace and user defined characters as delimiters. As an example show how the String Tokenizer class allows one to extract a sequence of tokens from a string with user defined delimiters.

b) Data File Handling

Need for Data file, Input Stream, Output Stream, Byte Stream (FileInputStream and FileOutputStream), Character Stream (FileReader, FileWriter), Operations Creation, Reading, Writing, Appending and Searching.

12. Recursion

Concept of recursion, simple recursive methods (e.g. factorial, GCD, binary search, conversion of representations of numbers between different bases).

Many problems can be solved very elegantly by observing that the solution can be composed of solutions to 'smaller' versions of the same problem with the base version having a known simple solution. Recursion can be initially motivated by using recursive equations to define certain methods. These definitions are fairly obvious and are easy to understand. The

definitions can be directly converted to a program. Emphasize that any recursion must have a base case. Otherwise, the computation can go into an infinite loop.

13. Implementation of algorithms to solve problems

The students are required to do lab assignments in the computer lab concurrently with the lectures. Programming assignments should be done such that each major topic is covered in at least one assignment. Assignment problems should be designed so that they are sufficiently challenging and make the student do algorithm design, address correctness issues, implement and execute the algorithm in Java and debug where necessary.

Self-explanatory.

14. Packages

Definition, creation of packages, importing user defined packages, interaction of objects across packages.

Java Application Programming Interface (API), development of applications using user defined packages.

15. Trends in computing and ethical issues

a) Artificial Intelligence, Internet of Things, Virtual Reality and Augmented Reality.

Brief understanding of the above and their impact on Society.

b) Cyber Security, privacy, netiquette, spam, phishing.

Brief understanding of the above.

c) Intellectual property, Software copyright and patents and Free Software Foundation.

Intellectual property and corresponding laws and rights, software as intellectual property.

Software copyright and patents and the difference between the two; trademarks; software licensing and piracy. free Software Foundation and its position on software, Open Source Software, various types of licensing (e.g. GPL, BSD).

Social impact and ethical issues should be discussed and debated in class. The important thing is for students to realise that these are complex issues and there are multiple points of view on many of them and there is no single 'correct' or 'right' view.

PAPER II: PRACTICAL – 30 MARKS

This paper of three hours duration will be evaluated internally by the school.