Maharishi International University 1971-1995

MAHARISHI UNIVERSITY OF MANAGEMENT

Professional Excellence and Higher Consciousness

CS-390:
Fundamental Programming Practices

Discovering the Structuring Principles of Creation

Dr. Paul Corazza

2019

Maharishi's Twelfth Year of Global Raam Raj

Maharishi University of Management is an Equal Opportunity Institution.

CS-390: Fundamental Programming Practices

Week		Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
1	AM	**Lesson 1:** Introduction to Java and the Eclipse Development Environment	**Lesson 2** (continued)	**Lesson 3:** Objects And Classes	**Lesson 3** (continued)	**Lesson 4:** Inheritance and Polymorphism	**Lesson 4** (continued)
	PM	**Lesson 2**: Fundamental Programming Structures in Java Lab	Lab	Lab **Lab 2 is due**	Lab	Lab **Lab 3 is due**	
2	AM	**Lesson 4** (continued)	**Lesson 5**: Building GUIs in Java with Swing	**Lesson 6:** Inner Classes	**Lesson 7:** Recursion	*Review for Midterm*	**Midterm**
	PM	Lab	Lab **Labs 4 is due**	Lab **Lab 5 is due**	Lab **Lab 6 is due**	*Study for Midterm*	
3	AM	**Lesson 8:** Lists	**Lesson 8** (Continued)	**Lesson 9:** Stacks and Queues	**Lesson 10**: Binary Search Trees	**Lesson 11**: Hashtables	**Lesson 12:** Exception Handling
	PM	Lab **Lab 7 is due**	Lab	Lab **Lab 8 is due**	Lab **Lab 9 is due**	Lab **Lab 10 is due**	
4	AM	**Lesson 13:** I/O, Databases	*Go over lab solutions*	*Review for Final*	**Final Exam**		
	PM	Lab **Labs 11, 12 are due**	*Review for Final* **Lab 13 is due**	*Study for Final*	**2-4 PM Standard Programming Exam**		

CS 390: Fundamental Programming Practices

Discovering the Structuring Principles of Creation

Main Objectives of FPP

The FPP course was created to fill gaps in the background of students when they first start their MSCS program; gaps of this kind have been classified into five areas. If you are in the FPP course, it means that the best first step you can take in this program is to strengthen your skills in these areas. The course will help you to:

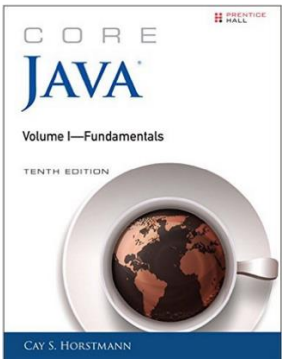
- Develop skills as a Java developer [we will evaluate those skills on the Standard Exam and in labs]
- Develop facility in the object-oriented paradigm [we will evaluate this aspect of learning in the midterm, final, and quizzes]
- Understand the principles behind optimal use of data structures, together with key points about optimal implementation and use in the Java language [we will evaluate this aspect of learning in the midterm and final and in one question on the Standard Exam]
- Become skillful in using the technique of recursion [we will evaluate this skill on the final exam]
- Significantly enhance problem-solving skills [this educational outcome will be tested in labs, midterm, final, and Standard Exam]

Class Schedule

Class is in session from 10 AM to 12:30 every weekday morning, with the final 15 minutes devoted to a group meditation, and from 1:30 to 3:05 every afternoon, with the final 20 minutes for group meditation. On Saturday, we meet only in the morning and follow the usual weekday format during the morning.

Textbooks

The *strongly recommended* textbook for the course is *Core Java 10th edition*, by Cay Horstmann, available through Amazon Books and Barnes and Noble (used copies are available at reasonable prices). One topic we will cover that is not in Volume 1 is Java I/O – this is covered in Chapter 2 of Volume 2; this chapter will be provided to you free of charge.



From Volume 2:

Chapter 2. Input and Output

In this chapter

- 2.1 Input/Output Streams
- 2.2 Text Input and Output
- 2.3 Reading and Writing Binary Data
- 2.4 Object Input/Output Streams and Serialization
- 2.5 Working with Files
- 2.6 Memory-Mapped Files
- 2.7 Regular Expressions

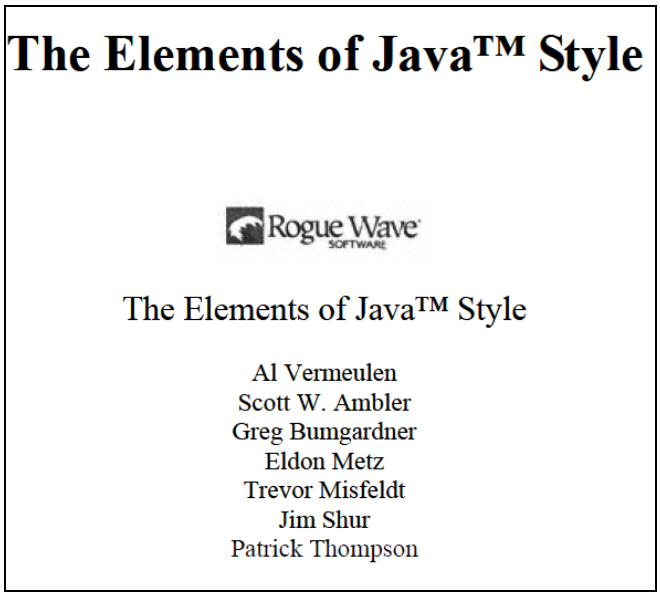
Readings

The *strongly recommended* text will supplement the material given in the lectures. You will find it helpful (and sometimes *necessary*) to read material from relevant sections in order to complete your understanding of the material and to assist you in doing the labs.

Supplementary Readings (not required)

1. Bloch, *Effective Java*, 3rd ed., 2017

2. *The Elements of Java Style*, Scott Ambler. We are using portions of this book in this course for one lab. It will be provided for you free of charge.



Labs

I will assign labs every day. Some will be in the form of pencil and paper exercises; others will require you to write Java programs. In class I will give details concerning how assignments will be evaluated. Labs **will count 10% of your final grade**. Labs are evaluated on an individual basis. If you work with someone else, make sure you understand your own answers – on the exams, the same concepts will reappear. Due dates for labs appear on the Calendar for the course and in the Assignments section on Sakai.

Exams

There will be two exams in the class. The following table shows the relative weights of each of the exams, along with other details:

Exam Number	Date Administered	Exam Content	Value
1	2nd Saturday	Lessons 1 – 6	40 %
2	Last Day	Lessons 7 - 13	45 %

Morning Meditation Bonus Points

Students are expected to attend 60% of the morning meditation sessions. Students who attend significantly more frequently than this will be awarded extra points according to the following table:

70% and above: .5% EC (16 days in a standard block)
80% and above: 1% EC (18 days in a standard block)
90% and above: 1.5% EC (20 days in a standard block)\

Academic Honesty

Students are expected to submit only their own work (except for labs or other activities designated as group activities). Duing exams, they must not look at other students' work, discuss exam contents with other students at any time (including bathroom breaks), or attempt to access outside resources (such as internet or email). The academic dishonesty policy stated on the Compro website is reproduced here:

> **Academic Dishonesty:** Graduate students caught cheating will receive a grade of NC. A second case of cheating results in suspension from the university. Cheating includes copying from someone else as well as letting someone else copy your materials, or not following the policies during the test (e.g., not using a cell phone at any time; not having notes, etc).

Grading

Your final grade will be a combination of your scores on Exams, Labs, Quizzes, and Professional Etiquette. Combined Exam scores count 85%; Lab scores count 10%; and Quizzes count 5%. The highest score you can receive for Professional Etiquette is 0; if you do poorly in the areas of attendance, attitude, or professional appearance, you may lose up to 3 points (3%) in your total score for the course.

Final Programming Test

- A two-hour programming test will be given on the afternoon of the last Thursday of class. The test will have two programming problems of easy/medium-level difficulty (by comparison with the FPP labs).
- The programming test will cover data structures and OO programming. Later in the course, I will provide you with more details about the types of questions you can expect on the programming test.
- The programming test will be graded on a Pass/Fail basis by the team in charge of the FPP/MPP program. Students who pass the test will get their grade on the basis of midterm, final, and other evaluations discussed above. Students who do not pass the test will not be allowed to continue on to MPP but, at the discretion of the teacher, may be allowed to repeat FPP.
- The highest grade a student who fails the FPP programming test may receive is "C+".

Reason for the Programming Test. One of the most important objectives of FPP for students is to be able to write Java programs to solve intermediate-level problems. Students need to be able to write functioning programs and to be able to debug them effectively enough to remove compiler and runtime errors. When students have managed once in a while to graduate from FPP without having developed these skills, the result has been that these students continually struggle to catch up in their later courses, and those courses are slowed down considerably because professors need to teach material and skills that were supposed to be covered in FPP. The programming test is a way of ensuring that students' knowledge and skill level have met the necessary standard. The exam is standardized to prevent any unfair bias—every FPP instructor during a block will administer the same standardized test.

Evaluation Modality	Value
Exams	85%
Labs	10%
Quizzes	5%
Professional Etiquette	-3 - 0 %
Standardized Programming Test	(depends)

Grading Scale

Range	Letter Grade
93-100	A
90 - 92	A-
87 - 89	B+
83 - 86	B
80 - 82	B-
77 - 79	C+
73 - 76	C
70 - 72	C-
0 - 69	NC

CS390: Fundamental Programming Practices
Discovering the Structuring Principles of Creation

Lesson 1: Introduction to Java
And the Eclipse Development Environment

Java is an object-oriented highly portable programming language that arose as an easy alternative to the once dominant, but error-prone, C++ language. Eclipse is one of many open source, powerful but easy-to-use integrated development environments for use with Java and related technologies. Working from deeper levels of intelligence allows one to accomplish more with fewer mistakes and less effort.

1. Java is an object-oriented programming language that is easier to use, less error-prone, and more portable (and nowadays, more popular) than C++. Java code is compiled to *bytecode*, which can then be converted to native code on a target platform, by way of a JVM interpreter. The inefficiency of an interpreter has largely been eliminated through the use of the *just-in-time compiler*, which compiles frequently occurring bytecode sequences to native code and caches these for further use, at runtime. Any language has the power to reveal or obscure the truth – as Maharishi says in SCI, "it is the power of speech that it can bind the boundless."

2. Eclipse is a leading, open-source, 100% Java, integrated development environment, which provides excellent support for editing, compiling, running, and debugging Java applications. By analogy, to create a good life, we need to handle inner life and at the same time, structure a life-supporting environment – the goal is to live 200% of life.

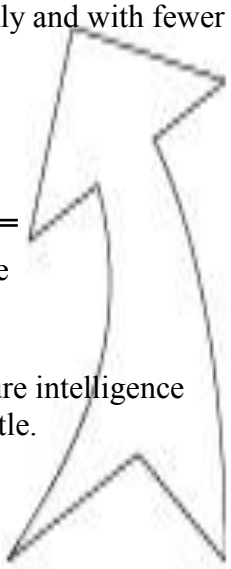
CONNECTING THE PARTS OF KNOWLEDGE
WITH THE WHOLENESS OF KNOWLEDGE

From pure intelligence to Java to IDE

1. Using Java, highly functional applications can be built more quickly and with fewer mistakes than is typically possible using C or C++.
2. To optimize the use of Java's features, IDE's such as Eclipse ease the work of the developer by handling in the background many routine tasks.

---

3. **Transcendental Consciousness**: To be successful, action must be based on the field of pure intelligence, which is located at the source of thought.
4. **Wholeness moving within Itself**: In Unity Consciousness, the pure intelligence located in TC is found pervading all of creation, from gross to subtle.



CS390: Fundamental Programming Practices
Discovering the Structuring Principles of Creation

Lesson 2: Fundamental Programming Structures In Java

Java is an object-oriented programming language that supports both primitive and object data types. These data types make it possible to store data in memory and modify it or perform computations on it to produce useful output. Execution of a program is an example of the "flow of knowledge"—the intelligence that has been coded into the program has a chance to be expressed when the program executes.

Maharishi's Science of Consciousness locates three components to any kind of knowledge: the knower, the object of knowledge, and the process of knowing. These can be found in the structure of a Java program: the "knower" aspect of the program is the intelligence underlying the creation of Java objects—a Java *class.* The *data* that a program works on, which is stored in program variables of either primitive or object type, is the "object of knoweldge." And the Java methods, which act on the data, are the "process of knowing."

1. Variables in Java are *declared* and *initialized* to provide room in RAM for the data that is to be stored. Pure consciousness manifests as individuals in space.

2. Variables of primitive type can be combined to form expressions through the use of *operators.* The Java syntax requires one to observe rules for forming expressions – precedence rules, type conversion rules, and others. Pure consciousness, likewise, also has laws that govern its self-combining. Combining of the three fundamental aspects of consciousness in all possible ways results in the manifest creation.

3. *Control flow* is supported in Java via the *if..else, for, while, do..while*, *switch,* and *for each* language elements. Loops are the CS analogue to the self-referral performance at the basis of all creation, whereas branching logic mirrors the tree-like hierarchy of natural laws that guide the activity in each layer of creation.

4. *Arrays* in Java support storage of multiple objects of the same type. Java supports multi-dimensional and ragged arrays; array copy and sort functions (accessible through the System and Arrays classes); and supports convenient forms of declaration and initialization. All CS data structures mirror the "existence" aspect of consciousness – the nervous system – whereas the *contents* of these structures mirrors the "intelligence" aspect; the pure potentiality of a data structure is as if brought to life by filling it with real data.

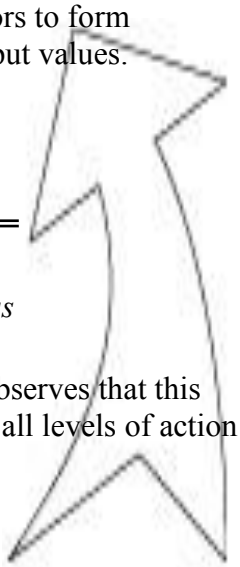
CONNECTING THE PARTS OF KNOWLEDGE
WITH THE WHOLENESS OF KNOWLEDGE

From expressions to Java programs

1. In Java, variables of primitive type can be combined using operators to form expressions, which may be evaluated to produce well-defined output values.
2. On a broader scale, objects in Java are "combined" by way of "messages" between objects, which collectively result in the behavior of a Java application.

---

3. **Transcendental Consciousness**: Pure consciousness is the field beyond type and interaction; it is the field of *unbounded awareness* and *infinite silence.*
4. **Wholeness moving within Itself**: In Unity Consciousness, one observes that this unbounded silent quality of awareness is spontaneously present at all levels of action in the world, and not just relegated to the transcendental field.



CS390: Fundamental Programming Practices
Discovering the Structuring Principles of Creation

Lesson 3: Objects and Classes

In the OO paradigm of programming, execution of a program involves objects interacting with objects. Each object has a type, which is embodied in a Java *class*. The intelligence underlying the functioning of any object in a Java program resides in its underlying class, which is the silent basis for the dynamic behavior of the objects. Likewise, pure consciousness is the silent level of intelligence that underlies all expressions of intelligence in the form of thoughts and actions in life.

1. The OO paradigm is a shift from old design and programming styles which are focused on machine-centric language models. In the OO paradigm, the focus shifts to mapping real world objects and dynamics to software objects and behavior; this parallel structure has proven to be more robust, less error prone, more scalable, and more cost-effective. In SCI we see that a more profound paradigm is discovered when the point of reference moves from the individual to the unbounded level – this is the CC paradigm in which self-sufficiency is based on true knowledge of the Self as universal, rather than the view of the self as a separate individual.

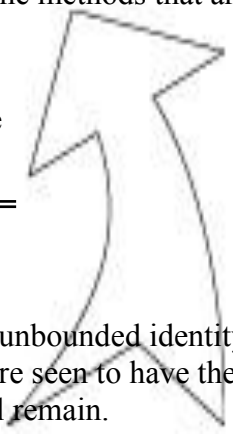
2. Static fields and methods are fields and methods whose lifetime persists throughout execution of the application, and when used with the public keyword, are globally accessible. The notion of "static" parallels the recognition that there is a field in life that is globally available and is always located in the same place in “memory”: namely, pure consciousness.

3. Java method calls are in every case *call by value* (and never *call by reference*). Even though an object reference can be passed into a method, the variable that stores the reference cannot be made to point to a different reference within the method. Therefore, only a *copy* of such a variable is ever passed to a method (in other words, call by value). Call by value is reminiscent of the incorruptible quality of pure consciousness – "fire cannot burn it, nor water wet it".

CONNECTING THE PARTS OF KNOWLEDGE
WITH THE WHOLENESS OF KNOWLEDGE

Object identity and identifying with unboundedness

1. A Java class specifies the type of data and the implementation of the methods that any of its instances will have.
2. Every object has not only state and behavior, but also identity, so that two objects of the same type and having the same state can be distinguished.



---

3. **Transcendental Consciousness**: TC is the identity of each individual, located at the source of thought.
4. **Wholeness moving within Itself**: In Unity Consciousness, one's unbounded identity is recognized to be the final truth about every object. All objects are seen to have the same ultimate identity, even though differences on the surface still remain.

CS390: Fundamental Programming Practices
Discovering the Structuring Principles of Creation

Lesson 4: Recursion

Computation of a function by recursion involves repeated self-calls of the function. Recursion is implicit also at the design level when a reflexive association is present. Recursion mirrors the self-referral dynamics of consciousness, on the basis of which all creation emerges. Recall: "Curving back on my own nature, I create again and again." (Gita, 9.8).

1. Java supports the creation of recursive methods, characterized by the fact that they call themselves in their method body. A self-calling method is a legitimate recursive function if it contains a *base case* -- a branch of code that exits the method under certain conditions but does not involve a self-call – and if the sequence of self-calls, on any input to the method, always converges to the base case. Likewise, a quest for self-knowledge not grounded in the direct experience of pure consciousness is baseless. (*Yastannah Veda Kimricha Karishyati* – What can the richas accomplish for him whose awareness is not established in That?)

2. When a recursion involves many redundant computations, one tries to write an iterative version of the method (using loops). Likewise, though the self-referral dynamics of consciousness provide a field for solving all problems, in practice it is often necessary to carry out many steps to implement a solution or handle a challenge.

CONNECTING THE PARTS OF KNOWLEDGE
WITH THE WHOLENESS OF KNOWLEDGE

Recursion creates from self-referral activity

1. In Java, it is possible for a method to call itself.
2. For a self-calling method to be a legitimate recursion, it must have a base case, and whenever the method is called, the sequence of self-calls must converge to the base case.

---

3. **Transcendental Consciousness**: TC is the self-referral field of existence, at the basis of all manifest existence.
4. **Wholeness moving within Itself**: In Unity Consciousness, one sees that all activity in the universe springs from the self-referral dynamics of wholeness. The "base case" – the reference point – is always the Self, realized as Brahman.

CS390: Fundamental Programming Practices
Discovering the Structuring Principles of Creation

Lesson 5: Building GUIs in Java with Swing

Swing is a windowing toolkit that allows developers to create GUIs that are rich in content and functionality. The ultimate provider of tools for the creation of beautiful and functional content is pure intelligence itself; all creativity arises from this field's self-interacting dynamics.

1. Swing classes are of two kinds: *components* and *containers.* A screen is created by creating components (like buttons, textfields, labels) and arranging them in one or more containers. Components and containers are analgous to the *manifest* and *unmanifest* fields of life; manifest existence, in the form of individual expressions, lives and moves within the unbounded container of pure existence.

2. Components are arranged in a container through the use of *layout managers* that organize components in different ways. FlowLayout preserves the size of components and lays components out horizontally, from left to right. BorderLayout lays out components in five positions – north, south, east, west and center; to preserve the size of components, BorderLayout is used in conjunction with FlowLayout. Likewise, all of manifest life is conducted by a vast network of natural laws.

3. Because containers are themselves a certain type of component, containers can be organized inside of other containers. Attractive visual design of GUIs is accomplished in Swing through the creative use of multiple layouts of container classes. The natural order of existence is created and maintained by the hidden dynamics of pure intelligence.

4. A GUI becomes responsive to user interaction (for example, button clicks and mouse clicks) through Swing's event-handling model in which event sources are associated with listener classes, whose actionPerformed method is called (and is passed an event object) whenever a relevant action occurs. To make use of this event-handling model, the developer defines a listener class, implements actionPerformed, and, when defining an event source (like a button), registers the listener class with this event source component. The "observer" pattern that is used in Swing mirrors the fact that in creation, the influence of every action is felt everywhere; existence is a field of infinite correlation; every behavior is "listented to" throughout creation.

CONNECTING THE PARTS OF KNOWLEDGE
WITH THE WHOLENESS OF KNOWLEDGE

The self-referral dynamics arising from the reflexive association of container classes

1. In Swing, components are placed and arranged in container classes for attractive display.
2. In Swing, containers are also considered to be components; this makes it possible to place and arrange container classes inside other container classes. These self-referral dyanmics support a much broader range of possibilities in the design of GUIs.

---

3. **Transcendental Consciousness**: TC is the self-referral field of all possibilities.
4. **Wholeness moving within Itself**: In Unity Consciousness, all activity is appreciated as the self-referral dynamics of one's own Self.

CS390: Fundamental Programming Practices
Discovering the Structuring Principles of Creation

Lesson 6: Inner Classes

Inner and nested classes allow classes to play the roles of instance variable, static variable and local variable, providing more expressive power to the Java language. Likewise, it is the hidden, unmanifest dynamics of consciousness that are responsible for the huge variety of expressions in the manifest world.

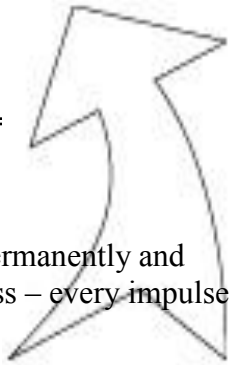
1. Classes are the fundamental notion in Java – programs are built from classes. With nested classes, Java makes it possible for this fundamental construct to play the roles of instance variable (member inner classes), static variable (static nested classes), and local variable (local inner classes). Likewise, in the unfoldment of creation, pure intelligence assumes the role of creative intelligence – in all of creation we find pure intelligence in the guise of individual expressions, individual existences, assuming diversified roles.

2. Inner classes – a special kind of nested class -- have access to the private members of their enclosing class, and the most commonly used kind of inner class is a *member* inner class. Likewise, when individual awareness is awake to its fully expanded, self-referral state, the memory of its eternal and infinitely dynamic nature becomes lively.

CONNECTING THE PARTS OF KNOWLEDGE
WITH THE WHOLENESS OF KNOWLEDGE

Inner classes retain the memory of their "unbounded" context

1. A *nested class* is a class that is defined inside another class.
2. An *inner class* is a nested class that has full access to its context, its enclosing class.



---

3. **Transcendental Consciousness**: TC is the unbounded context for individual awareness.
4. **Wholeness moving within Itself**: When individual awareness is permanently and fully established in its transcendental "context" – pure consciousness – every impulse of creation is seen to be an impulse of one's own awareness.

CS390: Fundamental Programming Practices
Discovering the Structuring Principles of Creation

Lesson 7: Inheritance, Interfaces and Polymorphism

Java supports inheritance between classes in support of the OO concepts of inherited types and polymorphism. Interfaces support encapsulation, play a role similar to abstract classes, and provide a safe alternative to multiple inheritance. Likewise, relationships of any kind that are grounded on the deeper values at the source of the individuals involved result in fuller creativity of expression with fewer mistakes.

1. One class (the *subclass)* inherits from another class (the *superclass*) if all proteced and public data and methods in the superclass are automatically accessible to the subclass, even though the subclass may have additional methods and data not found in the superclass. Java supports this notion of inheritance. In Java syntax, a class is declared to be a subclass of another by using the *extends* keyword. Likewise, individual intelligence "inherits from" cosmic intelligence, though each "implementation" is unique.

2. As a matter of good design, a class C should not be made a subclass of a class D unless C "IS-A" D. Likewise, individual intelligence "is" cosmic intelligence, though this relationship requires time to be recognized as true.

3. When a method is called on a subclass, the JVM by default uses *dynamic binding* to determine the correct method body to execute. Early binding (and hence a slight improvement in performance) can be forced by declaring a method final. In a similar way, it is said (Maharishi, *Science of Being*) that an enlightened individual need not continually plan and prepare in order to meet the needs of his daily life – instead, the enlightened enjoys spontaneous support of nature, and sees what to do as situations arise. Such individuals are an analogue to "late binding".

4. All classes in Java belong to the inheritance hierarchy headed by the `Object` class. Likewise, all individual consciousnesses inherit from the single unified field.

5. The classes in the Java reflection package can be used to construct an instance of a class (with parameters) from a String (which stores the name of the class) or Class object; similarly, it is possible to invoke a method on another class (with parameters) simply by knowing the String name of the method. Likewise, reflection on the infinite creative power of consciousness reveals the truth of every thing and gives rise to the creation of any object.

6. Interfaces are used in Java to specify publicly available services in the form of method declarations. A class that implements such an interface must make each of the methods operational. Interfaces may be used polymorphically, in the same way as a superclass in an inheritance hierarchy. Because many interfaces can be implemented by the same class, interfaces provide a safe alternative to multiple inheritance. The concept of an interface is analogous to the creation itself – the creation may be viewed as an “interface” to the undifferentiated field of pure consciousness; each object and avenue of activity in the creation serves as a reminder and embodiment of the ultimate reality.

CONNECTING THE PARTS OF KNOWLEDGE
WITH THE WHOLENESS OF KNOWLEDGE

Using Reflection to create objects at the level of "name"

1. Ordinarily, an object of a certain type is created in Java by calling the constructor of the class that realizes this type. This is object construction on the level of *form.*
2. Java's Reflection API allows the developer to construct on object based on the knowledge of the name (and the number and types of arguments required by the constructor). This is object construction on the level of *name.*

---

3. **Transcendental Consciousness**: The fundamental impulses that structure both the name and form of an object have their basis in the silent field of pure consciousness.
4. **Wholeness moving within Itself**: In Unity Consciousness, the finest structuring mechanics of creation are appreciated as modes of vibration of the Self.

CS390: Fundamental Programming Practices
Discovering the Structuring Principles of Creation

Lesson 8: Lists

The List ADT is one of the most general data types, capable of supporting most needs for storing a collection of objects in memory. Different implementations of this data type provide optimizations for different operations – such as insert, delete, find – that are typically supported by Lists. Lists give expression to the natural tendency of pure intelligence to express itself through a sequential unfoldment.

1. An Array List encapsulates the random access behavior of arrays, and incorporates automatic resizing and optionally may include support for sorting and searching. Random and sequential access provide analogies for forms of gaining knowledge. Knowledge by way of the intellect is always sequential, requiring steps of logic to arrive at an item of knowledge. Knowing by intuition (*prathibha*) , or by way of *ritam-bhara pragya*, is knowing the truth without steps – a kind of "random access" mode of gaining knowledge.

2. The List ADT captures the abstract notion of a "list"; it specifies certain operations that any kind of list should support (for example, *find, findKth, insert, remove*), without specifying the details of implementation. Different concrete implementations of this abstract data type (such as Array Lists and Linked Lists) meet the contract of the List ADT using different implementation strategies. Likewise, pure awareness is an abstraction of individual awareness; each individual provides a specific, concrete realization of unbounded and unmoving pure awareness.

3. A Linked List is another implementation of the List ADT in which efficiency of insert and remove operations are optimized, but eliminates support for random access of list elements. Insertions are done by rearranging links at the front of the list. Elements are located by traversing links, starting from the head of the list; a removal is then done by rearranging links. This difference between these two implementations of the List ADT recalls the nature of all differences – each item of creation has its beauty and its thorns. The full potential of each item is not found in the realm of relative differences but in the unbounded absolute level of existence. When this unbounded level is lived at the level of differences, the glory of the individual value is maximized.

CONNECTING THE PARTS OF KNOWLEDGE
WITH THE WHOLENESS OF KNOWLEDGE

All knowledge contained in point

1. An implementation of the abstract class `AbstractSequentialList` in Java (such as a `LinkedList`) results in a list that has only *sequential access* to its elements.
2. An implementation of the `RandomAccess` interface in Java (such as `ArrayList` and `Vector`) results in a list that has *random access* (and therefore, effectively, *instantaneous* access) to its elements.

---

3. **Transcendental Consciousness**: TC is the home of all knowledge. All knowledge has its basis in the unbounded field of pure consciousness.
4. **Wholeness moving within Itself**: In Unity Consciousness, when the home of all knowledge has become fully integrated in all phases of life, it is possible to know anything, any particular thing, instantly.

CS390: Fundamental Programming Practices
Discovering the Structuring Principles of Creation

Lesson 9: Stacks and Queues

Stacks and Queues are, essentially, special kind of lists with a highly restricted interface that permit rapid insertion and rapid access to elements, according to a "last in, first out" (Stacks) or "first in, first out" (Queues) scheme. These data structures express the MVS principle that creation emerges in the collapse of infinity to a point.

1. The Stack ADT is a special ADT that supports insertion of an element at “the top” and the removal of the top element, by way of operations *push* and *pop*, respectively. Similarly, the Queue ADT is a special ADT that supports insertion of an element at “the rear” (called *enqueuing*) and removal of an element from the “front” (called *dequeuing*). Both ADTs, when implemented properly, are extremely efficient. Sun (now Oracle) provides a Stack class and a Queue interface in its Collections API.

   Stacks and Queues make use of the MVS principle that the dynamism of creation arises in the concentration of dyamic intelligence to a point value ("collapse of infinity to a point"); stacks and queues achieve their high level of efficiency by concentrating on a single point of input (top of stack or rear of queue) and a single point of output (top of stack or front of queue).

CONNECTING THE PARTS OF KNOWLEDGE
WITH THE WHOLENESS OF KNOWLEDGE

Collapse of infinity to a point embodied in Stacks and Queues

1. Lists may be used as an all-purpose collection class. Nearly any need for storing collections of objects can be met by using some kind of list, though in some cases, other choices of data structures could improve performance. Lists have a more "unbounded" range of applicability.
2. Stacks and Queues are extremely specialized data structures, designed to accomplish (primarily) two operations with optimum efficiency. These data structures have a restricted range of applicability that is like a "point".

---

3. **Transcendental Consciousness**: Transcendental Consciousness is the unbounded value of awareness.
4. **Wholeness moving within Itself**: In Unity Consciousness, creation is seen as the interaction of unboundedness and point value: the unbounded collapses to its point value; point value expands to infinity; all within the wholeness of awareness.

CS390: Fundamental Programming Practices
Discovering the Structuring Principles of Creation

Lesson 10: Binary Search Trees

Binary Search Trees arose as a natural solution to the need for incorporating efficient insertion and deletion capabilities of linked lists with the support provided for fast sorting and binary search of sorted elements available in arrays and array lists. Expansion from a linear structure to a two-dimensional structure makes a solution of this kind possible. Likewise, any problem becomes easier to solve if one can transcend the boundaries of the problem.

1. A *binary search tree* (BST) is a binary tree in which the BST Rule is satisfied:

   > At each node *N*, every value in the left subtree of *N* is less than the value at *N*, and every value in the right subtree of *N* is greater than the value at *N*.

   BSTs provide efficient search, insert, and remove operations on orderable data. A binary search tree is an example of the principle of Diving: Because the structure is right, the basic operations are accomplished with maximum efficiency.

CONNECTING THE PARTS OF KNOWLEDGE
WITH THE WHOLENESS OF KNOWLEDGE

Fundamental patterns of consciousness in the realm of binary trees

1. Binary search trees support *insert, remove,* and *find* operations with efficient performance, as well as efficient support for accessing elements in order, such as *findMin*, *findMax*, and finding elements in a specified range.
2. The structure of complete binary trees involves an expansion from 1 to 2 to 4 to 8, and eventually to 64 (mirroring to a large extent the significant numbers that mark the progress of unfoldment with the Ved, from A to AK to the fourfold Rishi, Devata, Chhandas, Samhita, to 8-fold prakriti through the first Richa to the 64 fundamental impulses that structure the first 192 syllables of Rk Ved).

---

3. **Transcendental Consciousness**: TC is the home of all the impulses of natural law, of creative intelligence.
4. **Wholeness moving within Itself**: In Unity Consciousness, the emergence the structure of pure knowledge as appreciated as a self-referral activity of consciousness interacting with itself.

CS390: Fundamental Programming Practices
Discovering the Structuring Principles of Creation

Lesson 11: Hashtables

Hashtables provide even faster access to elements in a collection than a BST, but at the price of losing the sorted status of the elements. If maintaining order among the elements of a collection is not required, hashtables are the most efficient data structure for storing elements in memory, when insertion, deletion, and lookup operations are needed. Hashtables give concrete expression to the ability of pure intelligence to know any one thing instantaneously; this ability, or quality of intelligence, is known as *Ritam Bhara Pragya.*

1. The Hashtable ADT is a generalization of the concept of an array. It supports (nearly) random access of table elements by looking up with a (possibly) non-integer key. In the usual implementations, objects used as keys in a hashtable are "hashed", producing hashcode (a numeric value) and hashvalue (numeric value reduced in size to be less than the table size). The hashvalue is an index in an array that can be used to locate or insert an object. Hashtables illustrate the principle of *Do less and accomplish more* – they provide an incredibly fast implementation of the main List operations.

2. The most common implementation of hashtables uses *separate chaining;* each hashvalue is an index in an array of Lists; all objects with colliding hashvalue *i* are stored in the *i*th list in the array. The solution to the problem of avoiding hashvalue collisions, by using a List in each of the array slots, illustrates the *principle of the second element:* Using a list in each array slot provides a way to "harmonize" objects that were apparently in conflict because of identical hashvalues.

CONNECTING THE PARTS OF KNOWLEDGE
WITH THE WHOLENESS OF KNOWLEDGE

Random access expanded from integer index to arbitrary index,
from "point" to "infinity"

1. Arrays and ArrayLists provide highly efficient index-based access to a collection of elements.
2. The Hashtable ADT generalizes the behavior of an array by allowing non-integer keys (in fact, any object type can be used for a key), while retaining essentially random access efficiency for insertions, deletions, and lookups.

---

3. **Transcendental Consciousness**: TC is the home of all knowledge. The Upanishads declare "Know that by which all else is known" – this is the field of pure consciousness.
4. **Wholeness moving within Itself**: In Unity Consciousness, one sees that the "key" to accessing complete knowledge of any object is the infinite value of that object, pure consciousness, which is known in this state to be one's own Self. Knowing that level of the object, it then becomes possible to know any more relative level of the object as well.

CS390: Fundamental Programming Practices
Discovering the Structuring Principles of Creation

Lesson 12: Exception Handling in Java

Prior to the emergence of OO languages, error-handling was typically done using an unsystematic use of error codes. This approach led to confusion, programming errors, and costly maintenance. Java's exception-handling model (which is similar to those in other OO languages) systematizes the task of handling error conditions and integrates it with the OO paradigm supported by the language. This advance in programming practice illustrates the theme that "deeper knowledge has more profound organizing power."

1. Java's exception-handling model supports best practices in handling exceptions that arise during program execution. Likewise, establishing awareness in the home of all the laws of nature results in a life spontaneously lived in accord with natural law.

2. To use Exceptions effectively, when an Exception is thrown, a message should be *logged* so that the support team can review later; the Exception should be *thrown* up the call stack until a class that knows how to handle the Exception is reached; and this final class should *catch* and *handle* the Exception in an appropriate way (often, this means presenting an error message to the user). In a similar way, creation itself is structured in layers; the activity at each layer has its own unique set of governing laws; laws that pertain to one level or layer may no longer be applicable at another level.

3. Methods whose declaration includes a *throws* clause can be called from within another method only if the latter is declared with the same *throws* clause, or if a try/catch block is included to catch any of the declared exceptions that are thrown. This phenomenon is reminiscent of the Principle of Diving: once the initial conditions have been met, a correct dive into the depths occurs automatically. (The *throws* clause is the initial condition; the compiler then automatically requires additional coding in order to handle exceptions that may occur.)

4. In designing an application, a good practice is to design a small set of Exception classes that will be used to classify exceptions during runtime. Then, each runtime exception should be re-thrown as an instance of one of your custom-defined Exception classes. This illustrates the principle that "knowledge has organizing power" – when a deeper quality of intelligence is brought to bear on a situation, the situation spontaneously is re-organized so as to reflect a higher degree of orderliness.

CONNECTING THE PARTS OF KNOWLEDGE
WITH THE WHOLENESS OF KNOWLEDGE

Right action in the field of exception-handling

1. If a Java method has a *throws* clause in its declaration, the compiler requires the developer to (write code to) handle potential exceptions whenever the method is called.
2. To handle exceptions in the best possible way, *logging* should occur as soon as an exception is thrown, and the exception should be re-thrown up the call stack until a method belonging to a class with an appropriate set of responsibilities is reached – and within this method, the exception should be *caught* and handled.

---

3. **Transcendental Consciousness**: TC is the home of all the laws of nature, the home of "right action".
4. **Wholeness moving within Itself**: Action in the state of Unity Consciousness is spontaneoulsy right and uplifting to the creation as a whole.

CS390: Fundamental Programming Practices
Discovering the Structuring Principles of Creation

Lesson 13:
Working With Files and Databases

Java provides convenient tools for reading and writing files, and for accessing data stored in a database. The relationship between stored data and an executing program parallels the relationship between awareness and its interaction with the world; that interaction is most successful and rewarding if awareness is broad (corresponding to a well-designed program) and is well integrated with the laws of nature, with ways of manifest existence (JDBC).

1. Reading a File in Java is accomplished by using a FileReader (or Scanner). Writing to a File is accomplished by using a FileWriter. More generally, "input" in human life is handled by the *senses*; "output" is handled by the *organs of action.* Both have their source in the field of pure creative intelligence.

2. JDBC provides an API for interacting with a database using SQL. To interact efficiently with a database, you typically use the database vendor's driver that allows communication between the JVM and the database. This is reminiscent of the Principle of Diving – once the initial conditions have been met, a good dive is automatic. (Here, the initial conditions are correct configuration of the data source and code to load the database driver; once the set up is right, interacting with the database is "effortless".)

CONNECTING THE PARTS OF KNOWLEDGE
WITH THE WHOLENESS OF KNOWLEDGE

Expansion of consciousness leads to expanded territory of influence

1. Since Java is an OO language, it supports storage and manipulation of data within appropriate objects.
2. To work with real data effectively, Java supports interaction with external data stores (databases) through the use of various JDBC drivers, and the JDBC API.

---

3. **Transcendental Consciousness**: TC is the field of truth, the field of *Sat.* "Know that by which all else is known." -- *Upanishads*
4. **Wholeness moving within Itself**: In Unity Consciousness, the final truth about life is realized in a single stroke of knowledge.