Q. What are the differences between functions and procedure? (3 Points)

Functions	Procedures
A function has a return type and returns a value.	A procedure does not have a return type.
A function does not allow output parameters	A procedure allows both input and output parameters.
You can call a function using a select statement.	You cannot call a procedure using select statements.

Q. What are the 3 semantic models of parameter passing? (3 Points)

In Mode: Calling function with some parameter but not returning anything.

Out mode: Calling function with no parameter but returning a parameter.

In-out mode: Calling function with some parameter and returning a parameter.

Q. Define the generic subprogram and explain generic methods in Java. (2 * 2 Points) (Show Example)

A generic or polymorphic subprogram is one that takes parameters of different types on different activations.

Generic methods are methods that introduce their own type parameters. This is like declaring a generic type, but the type parameter's scope is limited to the method where it is declared. Static and non-static generic methods are allowed, as well as generic class constructors.

Q. Describe the shallow-access method of implementing dynamic scoping. (4 Points)

Shallow access is an alternative implementation method, not an alternative semantics. The semantics of deep access and shallow access are identical. In the shallow-access method, variables declared in subprograms are not stored in the activation records of those subprograms.

Q. How the Java way of parametrized abstract data types (2 Points)

- Generic parameters must be classes
- Most common generic types are the collection types, such as LinkedList and ArrayList

- Eliminate the need to cast objects that are removed
- Eliminate the problem of having multiple types in a structure

Q. Which version of Java introduced Generics (1 Point)

Java 5

Q. What advantage do monitor have over semaphores? (2 Points). Show some example (2 Points)

Advantage of monitor over semaphore Mutual exclusion in monitor is automatic while mutual exclusion in semaphore needs to be coded explicitly. Shared variables are global to all processes in the monitor while shared variables are hidden in semaphores.

Q. Explain the three reasons accessors to private types are better than making types public.

Encapsulation provides data hiding and more control on the member variables. If an attribute is public then anyone can access it and can assign any value to it. But if your member variable is private and you have provided a setter for it.
Then you always have an option to put some constraints check in the setter method to avoid setting an illogical value.

Q. Explain Object-Orientation

a) Describe the three characteristic features of Object-Oriented programming

Object Oriented languages are languages that encourage reusability and abstraction. Three important concepts in object Orientation is Inheritance, Encapsulation and Polymorphism

b) What is single inheritance?

Single inheritance is when a class extends one class, that is a base class extends only class (super class).

c) What is multiple inheritance?

Multiple inheritance is when a class extends more than one class, that is the class has multiple super classes.

d) What is polymorphic variable?

A polymorphic variable is a variable that has the same interface, but different implementation across classes

e) What is an overriding a method?

Overriding a method; is when a child class extends or overrides the implementation of its super class method

f) What is the message protocol of an object?

Collections of an object's method.

g) What is a nesting class?

A nested class is a class within a class, it can be instance or static class

h) What is an interface?

An interface provides definition of method interface without the implementation, which can further be extended

Q. Describe the actions of the three Java methods that are used to support cooperation synchronization. (3 Points) (Not Sure)

- To prevent thread interference.
- To prevent consistency problem.

Q. Describe exception handling in Java. (5 Points)

- **Exception** is an error or constraint check that occurs in a program which can stop the program from running.
- **Exception handler** handles exceptions if they occur, which prevent programs from failing.
- **Raising an exception** is a process of alerting or throwing an error or raising an exception if an event occurs.
- **Finally** is part of the exception handler that always runs no matter what the result of the exception handler is.

(Example copy from the MCQs :P)

Q. Describe the general characteristics of subprograms?

- Each subprogram has a single-entry point.
- The calling program unit is suspended during the execution of the called subprogram, which implies that there is only one subprogram in execution at any given time.
- Control always returns to the caller when the subprogram execution terminates

Q. What is given in the header of subprogram?

A subprogram header serves several purposes. First, it specifies that the following syntactic unit is a subprogram definition of some kind. In languages that have more than one kind of subprogram, the kind of the subprogram is usually specified with a special word. Second, if the subprogram is not anonymous, the header provides a name for the subprogram. Third, it may optionally specify a list of parameters. For example, in Python: **def add (a,b):** This is a header of a Python subprogram named add. Here, def implies that following line specifies a subprogram, a and b are parameters of the subprogram.

Q. What are formal and actual parameters?

Formal parameter — the identifier used in a method to stand for the value that is passed into the method or subprogram by a caller.
Actual parameter — the actual value that is passed into the method or subprogram by a caller.

Q. Explain pass-by-value parameter passing mode.

It means you are making a copy in memory of the actual parameter's value that is passed in, a copy of the contents of the actual parameter. Pass by value is used when you are only "using" the parameter for some computation, not changing it for the client program.

Q. Define Abstract Data Type.

The definition of ADT only mentions what operations are to be performed but not how these operations will be implemented. It does not specify how data will be organized in memory and what algorithms will be used for implementing the operations. It is called “abstract” because it gives an implementation-independent view. The process of providing only the essentials and hiding the details is known as abstraction.

Q. Discuss Abstract Data types in Java.

In Java abstract data type, we can only know what operations are to be performed and not how to perform them i.e. it does not tell how algorithms are to be implemented or how the data will be organized in the memory. This type is thus called abstract as it does not give the view of implementation.

Eg: **int** – We can do arithmetic operations, comparisons but do know how they are implemented. (Write a java code explaining int)

Q. What is the difference between a class and an instance variable?

Instance variables are declared in a class. They are created when an object is created with the use of the keyword 'new' and destroyed when the object is destroyed. **Class variables** also known as static variables are declared with the static keyword in a class. They are created when the program starts and destroyed when the program stops.

Q. Define overload subprogram and explain overloaded methods in Java. (2 * 2 Points) (Show Example)

Method Overloading is a feature that allows a class to have more than one method having the same name, if their argument lists are different. It is like constructor overloading in Java. Overloading can be achieved in three different ways in Java:

Number of parameters: subtract(int, int)
subtract(int, int, int)

Data type of parameters: subtract(int, int)
subtract(int, float)

Sequence of Data type of parameters: product(int, float)
product(float, int)

(Write different names please not the same)

Q. Why are properties (getter/setter) better than specifying an instance variable to the public?

The main difference between making a field public vs. exposing it through getters/setters is holding control over the property. If you make a field public, it means you provide direct access to the caller. Then, the caller can do anything with your field, either knowingly or unknowingly. For example, one can set a field to a null value, and if you use that field in another method, it may blow up that method with a null pointer exception.

Q. Define task, synchronization, competition and cooperation synchronization, Liveness and deadlock. (5 Points)

A **task** or process is a program unit that can be in concurrent execution with other program units. **Synchronization:** A mechanism that controls the order in which tasks execute. **Cooperation:** task A must wait for task B to complete some specific activity before task A can continue its execution, e.g., the producer-consumer problem. **Competition:** Two or more tasks must use some resource that cannot be simultaneously used, e.g., a shared counter. **Liveness** is a characteristic that a program unit may or may not have. In sequential code, it means the unit will eventually complete its execution. **Deadlock:** In a concurrent environment, a task can easily lose its liveness. If all tasks in a concurrent environment lose their liveness, it is called deadlock.

Additional Questions:

Q. From where java objects are allocated?

In Java, all objects are allocated from the heap and accessed through reference variables. Most objects are allocated with the new operator, but there is no explicit deallocation operator. Garbage collection is used for storage reclamation.

Q. What is a java package and what is its purpose?

Java includes a naming encapsulation construct: the package. Packages can contain more than one type definition, and the types in a package are partial friends of one another. Partial here means that the entities defined in a type in a package that either are public or protected or have no access specifier are visible to all other types in the package.

Object Orientation Programming

With object-orientation, programmers can begin with an existing abstract data type and design a modified descendant of it to fit a new problem requirement. The abstract data types in object-oriented languages are usually called classes. As with instances of abstract data types, class instances are called objects. A class that is defined through inheritance from another class is a derived class or subclass. A class from which the new class is

derived is its parent class or superclass. If a new class is a subclass of a single parent class, then the derivation process is called single inheritance. If a class has more than one parent class, the process is called multiple inheritance. The subprograms that define the operations on objects of a class are called methods. The calls to methods are called messages. The entire collection of methods of an object is called the message interface of the object. Abstract methods are the methods that are declared without an implementation. A class that includes at least one abstract method is called an abstract class. Such a class usually cannot be instantiated. Abstract methods get overridden in the descendant of the class it was declared. A method is said to be overridden, when the subclass modifies the behavior of its inherited (mentioned) method.