QUESTION BANK

U23CST34 — Object Oriented Programming

UNIT I
Introduction to OOP and Java

PART A 2-Mark Questions with Answers

Q1. Define Object Oriented Programming. *[CO1 | UNIT I | 2M]*

Answer:

OOP is a programming paradigm that organizes software around objects rather than functions.
An object bundles state (data/fields) and behavior (methods) together. Key principles are Encapsulation, Inheritance, Polymorphism, and Abstraction.

Q2. What are the four pillars of OOP? *[CO1 | UNIT I | 2M]*

Answer:

1. Encapsulation – hiding data within a class using access modifiers.
2. Inheritance – child class inherits properties from parent class.
3. Polymorphism – same method behaves differently in different contexts.
4. Abstraction – hiding complexity and exposing only essential features.

Q3. What is the difference between a class and an object? *[CO1 | UNIT I | 2M]*

Answer:

Class: A blueprint or template that defines fields and methods. It does not occupy memory by itself.
Object: A real-world instance of a class that occupies memory. Example: class Dog { } → Dog d = new Dog(); // d is an object.

Q4. List any four Java buzzwords. *[CO1 | UNIT I | 2M]*

Answer:

1. Simple – easy syntax, no pointers.
2. Platform Independent – compiled to bytecode; runs on any JVM (Write Once Run Anywhere).
3. Robust – strong type checking, exception handling, garbage collection.
4. Secure – no pointer arithmetic; sandboxed execution environment.

Q5. What is the role of the JVM? *[CO1 | UNIT I | 2M]*

Answer:

JVM (Java Virtual Machine) is an abstract machine that executes Java bytecode.
It provides platform independence by interpreting/JIT-compiling bytecode specific to the host OS. It also manages memory via the garbage collector.

Q6. What is a constructor in Java? *[CO1 | UNIT I | 2M]*

Answer:

A constructor is a special method that is automatically called when an object is created.
Rules: same name as the class, no return type. Types: default (no-arg) and parameterized.
Example: class Box { Box(int l) { length = l; } }

Q7. What are access specifiers in Java? *[CO1 | UNIT I | 2M]*

Answer:
Access specifiers control visibility of class members.
private – accessible only within the class.
default – within the same package.
protected – same package + subclasses.
public – accessible from anywhere.

Q8. Differentiate between static and non-static members. *[CO1 | UNIT I | 2M]*

Answer:
Static members belong to the class and are shared among all objects. Accessed via class name: ClassName.member.
Non-static (instance) members belong to each individual object. Accessed via object reference: obj.member.

Q9. What is method overloading? *[CO1 | UNIT I | 2M]*

Answer:
Method overloading is defining multiple methods with the same name but different parameter lists (different number or types of arguments) in the same class.
It is resolved at compile time (static binding). Return type alone cannot differentiate overloaded methods.

Q10. What is the use of the 'this' keyword? *[CO1 | UNIT I | 2M]*

Answer:
'this' is a reference variable that refers to the current object.
Uses: (1) Distinguish instance variables from local variables with the same name.
(2) Call another constructor in the same class: this().
(3) Pass current object as an argument to a method.

PART B 16-Mark Questions (Answer Space Provided)

Q1. Explain the features of Object Oriented Programming with examples. *[CO1 | UNIT I | 16M]*

(a) Describe Encapsulation with a real-world example and code. (4M)
(b) Explain Inheritance and Polymorphism with suitable Java programs. (6M)
(c) Illustrate Abstraction using abstract class and interface with examples. (6M)

Answer: _____

Q2. Describe the structure of a Java program and explain all primitive data types with their sizes. Write a Java program to demonstrate array operations (create, initialize, traverse, sort). *[CO1 | UNIT I | 16M]*

(a) Java program structure: package, import, class, main method. (4M)

(b) Primitive data types: byte, short, int, long, float, double, char, boolean with sizes and examples. (6M)

(c) Java program with array creation, initialization, traversal and sorting. (6M)

Answer: _____

Q3. Explain all types of operators in Java with examples. Also describe control statements with syntax and a program to demonstrate all loop types. *[CO1 | UNIT I | 16M]*

(a) Arithmetic, Relational, Logical, Bitwise, Assignment, Ternary operators with examples. (6M)

(b) if-else, switch, for, while, do-while with syntax and examples. (6M)

(c) Java program using all loop types to print patterns. (4M)

Answer: _____

Q4. Define a class 'Student' in Java with appropriate fields, constructor, and methods. Demonstrate object creation, access specifiers, and static members. *[CO1 | UNIT I | 16M]*

(a) Class definition with private fields, parameterized constructor, getters/setters. (6M)

(b) Static field to track number of objects created. (4M)

(c) Main method creating multiple objects and displaying data. (6M)

Answer: _____

Q5. Write a Java program to demonstrate constructor overloading and method overloading. Explain the concept of JavaDoc comments and static methods. *[CO1 | UNIT I | 16M]*

(a) Constructor overloading with 3 constructors and example. (6M)

(b) Method overloading with different parameter types. (4M)

(c) Static methods and JavaDoc comment syntax with example. (6M)

Answer: _____

UNIT II

Inheritance, Packages and Interfaces

PART A 2-Mark Questions with Answers

Q1. What is inheritance in Java? *[CO2 | UNIT II | 2M]*

Answer:

Inheritance is a mechanism by which a child class acquires the properties and behaviors of a parent class using the extends keyword.

It promotes code reusability and represents an IS-A relationship. Java supports single, multilevel, and hierarchical inheritance.

Q2. What is the use of the 'super' keyword? *[CO2 | UNIT II | 2M]*

Answer:

super refers to the immediate parent class object.

Uses: (1) super() – call parent constructor.

(2) super.method() – call overridden parent method.

(3) super.field – access hidden parent field.

Q3. Differentiate method overloading and method overriding. *[CO2 | UNIT II | 2M]*

Answer:

Overloading: Same class, same method name, different parameters; resolved at compile time (static binding).

Overriding: Parent-child classes, same method name and signature; resolved at runtime (dynamic binding). Requires @Override annotation.

Q4. What is dynamic method dispatch? *[CO2 | UNIT II | 2M]*

Answer:

Dynamic method dispatch (runtime polymorphism) is the mechanism by which a call to an overridden method is resolved at runtime.

When a parent reference holds a child object and calls an overridden method, the child's version executes. Example: Animal a = new Dog(); a.sound(); // calls Dog's sound().

Q5. What is an abstract class? *[CO2 | UNIT II | 2M]*

Answer:

An abstract class is a class declared with the abstract keyword that cannot be instantiated.

It can contain abstract methods (no body) that subclasses must override, and concrete methods.

If a class has any abstract method, it must be declared abstract.

Q6. What is an interface in Java? *[CO2 | UNIT II | 2M]*

Answer:

An interface is a blueprint of a class containing abstract methods and constants.
A class implements an interface using implements keyword and must provide body for all methods.
Interfaces achieve full abstraction and support multiple inheritance in Java.

Q7. Differentiate abstract class and interface. *[CO2 | UNIT II | 2M]*

Answer:
Abstract class: can have concrete methods, constructors, any type of fields; single inheritance.
Interface: methods are public abstract by default (Java 8+: default/static allowed); fields are public static final; supports multiple inheritance.

Q8. What is a package in Java? *[CO2 | UNIT II | 2M]*

Answer:
A package is a namespace that organizes related classes and interfaces into a single unit.
Benefits: avoids naming conflicts, provides access protection, and makes code modular.
Created with package keyword; imported with import keyword.

Q9. What is the final keyword in Java? *[CO2 | UNIT II | 2M]*

Answer:
final variable: value cannot be changed (constant).
final method: cannot be overridden by a subclass.
final class: cannot be subclassed (e.g., java.lang.String is a final class).

Q10. What are inner classes in Java? *[CO2 | UNIT II | 2M]*

Answer:
An inner class is a class defined within another class.
Types: (1) Non-static inner class – can access all outer members.
(2) Static nested class – can access only static outer members.
(3) Local inner class – defined inside a method.
(4) Anonymous inner class – no name, defined and used once.

PART B 16-Mark Questions (Answer Space Provided)

Q1. Explain the types of inheritance in Java with programs. Demonstrate single, multilevel, and hierarchical inheritance with appropriate examples. *[CO2|UNITII|16M]*
(a)Singleinheritance: class Animal → class Dog with example.(4M)
(b)Multilevelinheritance:Animal→Dog → Labrador with constructor chainingusingsuper(). (6M)
(c) Hierarchical inheritance: Animal → Dog, Animal→Catwith method overriding. (6M)
Answer: _____

Q2. Explain method overriding and dynamic method dispatch with a Java program. Also describe the use of abstract classes. *[CO2 | UNIT II | 16M]*

(a) Method overriding rules and @Override annotation with example. (4M)
(b) Runtime polymorphism: parent reference → child object; method resolution at runtime. (6M)
(c) Abstract class with abstract methods and concrete implementation in subclasses. (6M)

Answer: _____

Q3. Write a Java program to demonstrate interfaces including default methods, multiple interface implementation, and interface inheritance. *[CO2 | UNIT II | 16M]*

(a) Basic interface with abstract methods and implementing class. (4M)
(b) Multiple interfaces implemented by a single class. (6M)
(c) Java 8 default methods in interface with example and interface extending interface. (6M)

Answer: _____

Q4. Explain packages in Java. Create a user-defined package with classes and demonstrate importing it in another program. Also explain access protection with packages. *[CO2|UNITII|16M]*

(a) Creating a package and compiling with -d option. (4M)
(b) User-defined package 'com.college.student' with Student class. (6M)
(c) Importing packages and access protection rules across packages. (6M)

Answer: _____

Q5. Write a Java program that demonstrates the use of inner classes (non-static, static nested, local, and anonymous inner classes) with suitable examples. *[CO2 | UNIT II | 16M]*

(a) Non-static inner class accessing outer class members. (4M)
(b) Static nested class and local inner class examples. (6M)
(c) Anonymous inner class implementing an interface at runtime. (6M)

Answer: _____

UNIT III
Exception Handling and Multithreading

PART A 2-Mark Questions with Answers

Q1. What is an exception in Java? *[CO3 | UNIT III | 2M]*

Answer:
An exception is an abnormal event that disrupts the normal flow of program execution at runtime.
Exceptions are objects of classes derived from java.lang.Throwable. Java handles them using try, catch, finally, throw, and throws keywords.

Q2. Differentiate checked and unchecked exceptions. *[CO3 | UNIT III | 2M]*

Answer:
Checked exceptions: must be handled or declared using throws; checked at compile time. Examples: IOException, SQLException.
Unchecked exceptions: subclasses of RuntimeException; not required to be caught; checked at runtime. Examples: NullPointerException, ArrayIndexOutOfBoundsException.

Q3. What is the difference between throw and throws? *[CO3 | UNIT III | 2M]*

Answer:
throw: used to explicitly throw an exception object inside a method. Syntax: throw new ExceptionType(msg);
throws: declares that a method may throw one or more exceptions. Syntax: void method() throws IOException { }

Q4. What is the finally block? *[CO3 | UNIT III | 2M]*

Answer:
The finally block always executes regardless of whether an exception is thrown or caught.
It is used for cleanup operations like closing files, database connections, or releasing resources.
Structure: try { } catch(E e) { } finally { // always runs }

Q5. How do you create a user-defined exception? *[CO3 | UNIT III | 2M]*

Answer:
Extend the Exception class (for checked) or RuntimeException (for unchecked).
Example: class InvalidAgeException extends Exception { InvalidAgeException(String msg){ super(msg); } }
Throw it with: throw new InvalidAgeException("Age must be >= 18");

Q6. What is a thread in Java? *[CO3 | UNIT III | 2M]*

Answer:
A thread is the smallest unit of execution within a process. Java supports multithreading —

running multiple threads concurrently.
A thread has 5 states: New, Runnable, Running, Blocked/Waiting, and Terminated.

Q7. What are the two ways to create a thread in Java? *[CO3 | UNIT III | 2M]*

Answer:
1. Extending Thread class: class MyThread extends Thread { public void run(){ } }; call start().
2. Implementing Runnable interface: class Task implements Runnable { public void run(){ } }; new Thread(new Task()).start().
Runnable is preferred as the class can still extend another class.

Q8. What is synchronization in Java? *[CO3 | UNIT III | 2M]*

Answer:
Synchronization is a mechanism that restricts multiple threads from accessing a shared resource simultaneously to prevent race conditions.
A method or block is marked synchronized — only one thread can execute it at a time. It uses intrinsic locks (monitors).

Q9. What is the difference between wait() and sleep()? *[CO3 | UNIT III | 2M]*

Answer:
wait(): releases the lock, causes thread to wait until notify() is called; must be called in synchronized context.
sleep(): pauses the thread for a specified time (ms) without releasing the lock; called anywhere.

Q10. What is auto-boxing and unboxing? *[CO3 | UNIT III | 2M]*

Answer:
Auto-boxing: automatic conversion of a primitive to its wrapper class. Example: int x=5; Integer obj=x;
Unboxing: automatic conversion of wrapper class back to primitive. Example: Integer obj=42; int x=obj;
This allows primitives to be used in collections like ArrayList<Integer>.

PART B 16-Mark Questions (Answer Space Provided)

Q1. Explain exception handling in Java with try, catch, finally, throw and throws. Write programs to demonstrate multiple catch blocks, nested try, and custom exceptions.
[CO3 | UNIT III | 16M]

(a) Exception hierarchy: Throwable → Error / Exception → Checked / Unchecked. (4M)
(b) Multiple catch blocks and nested try-catch with examples. (6M)
(c) Creating and using a custom exception (InvalidAgeException). (6M)

Answer: _____

Q2. Describe the Java Thread model. Write Java programs to create threads using both Thread class and Runnable interface. Explain all thread lifecycle states. *[CO3 | UNIT III | 16M]*

(a) Thread lifecycle: New, Runnable, Running, Blocked, Terminated with diagram description. (4M)

(b) Creating thread by extending Thread class with example. (6M)

(c) Creating thread by implementing Runnable interface with example and comparison. (6M)

Answer: _____

Q3. Explain thread synchronization and inter-thread communication in Java. Write a program to demonstrate the Producer-Consumer problem using wait() and notify(). *[CO3 | UNIT III | 16M]*

(a) Race condition problem and synchronized keyword with example. (4M)

(b) Synchronized method and synchronized block with comparison. (6M)

(c) Producer-Consumer solution using wait(), notify(), shared buffer. (6M)

Answer: _____

Q4. Explain thread priorities, suspending, resuming and stopping threads. Write a program demonstrating multiple threads with different priorities and demonstrate Multithreading wrappers. *[CO3 | UNIT III | 16M]*

(a) Thread priority values (1-10), setPriority(), getPriority() with example. (4M)

(b) Safe thread suspension/stopping using volatile flag instead of deprecated methods. (6M)

(c) Wrapper classes and auto-boxing/unboxing with all primitive-wrapper pairs. (6M)

Answer: _____

Q5. Write a Java program to demonstrate a multi-threaded bank account system where multiple threads perform deposit and withdrawal operations, ensuring data integrity using synchronization. *[CO3 | UNIT III | 16M]*

(a) BankAccount class with synchronized deposit() and withdraw() methods. (6M)

(b) Multiple Thread objects simulating concurrent transactions. (6M)

(c) Output showing correct balance after all transactions. (4M)

Answer: _____

UNIT IV

I/O, Generics, String Handling

PART A 2-Mark Questions with Answers

Q1. What are streams in Java I/O? *[CO4 | UNIT IV | 2M]*

Answer:

A stream is a sequence of data that flows between a source and a destination.
Byte Streams (InputStream/OutputStream): handle raw bytes; used for binary data.
Character Streams (Reader/Writer): handle Unicode text; used for text files.

Q2. What is the difference between FileReader and BufferedReader? *[CO4 | UNIT IV | 2M]*

Answer:

FileReader reads one character at a time directly from the file — slow for large files.
BufferedReader wraps FileReader and reads a chunk into a buffer, improving efficiency. Provides readLine() to read entire lines at once.

Q3. What is generics in Java? *[CO4 | UNIT IV | 2M]*

Answer:

Generics allow classes, interfaces, and methods to operate on parameterized types, providing type safety at compile time.
Example: List<String> list = new ArrayList<>(); — ensures only String can be added.
Benefits: type safety, no casting required, code reusability.

Q4. What is a bounded type parameter in generics? *[CO4 | UNIT IV | 2M]*

Answer:

Bounded type parameters restrict the types that can be used as type arguments.
Upper bound: <T extends Number> — T can be Number or any subclass.
Wildcard: List<? extends Number> — accepts any subtype of Number.

Q5. What is type erasure in Java generics? *[CO4 | UNIT IV | 2M]*

Answer:

Type erasure is the process where the Java compiler removes all generic type information and replaces it with raw types (Object or bounded types) at compile time.
As a result, generic type information is NOT available at runtime. This is done for backward compatibility.

Q6. Differentiate String, StringBuffer and StringBuilder. *[CO4 | UNIT IV | 2M]*

Answer:

String: immutable; every modification creates a new object; thread-safe.

StringBuffer: mutable; synchronized (thread-safe); slower.
StringBuilder: mutable; not synchronized; fastest; preferred for single-threaded use.

Q7. What is the difference between == and equals() for Strings? *[CO4 | UNIT IV | 2M]*

Answer:
==: compares references (memory addresses) of two objects.
equals(): compares the content (characters) of two String objects.
Example: String s1="Hi"; String s2=new String("Hi"); s1==s2 is false; s1.equals(s2) is true.

Q8. Name any five String class methods with purpose. *[CO4 | UNIT IV | 2M]*

Answer:
1. length() – returns number of characters.
2. charAt(i) – returns character at index i.
3. substring(i,j) – extracts substring from index i to j-1.
4. toUpperCase() – converts all characters to uppercase.
5. split(regex) – splits string into an array based on delimiter.

Q9. What is try-with-resources in Java? *[CO4 | UNIT IV | 2M]*

Answer:
Try-with-resources (Java 7+) automatically closes resources (streams) after the try block.
Resources must implement AutoCloseable interface.
Example: try(BufferedReaderbr = new BufferedReader(new FileReader("f.txt"))) { // br closed automatically }

Q10. What is the difference between Generic class and Generic method? *[CO4 | UNIT IV | 2M]*

Answer:
Generic class: type parameter is specified at class level and applies to the whole class. Example: class Box<T> { T value; }
Generic method: type parameter is declared before return type and applies only to that method. Example: public <T> void print(T item) { }

PART B 16-Mark Questions (Answer Space Provided)

Q1. Explain Java I/O streams in detail. Write programs to read from and write to a file using both byte streams and character streams. Demonstrate try-with-resources. *[CO4 | UNIT IV | 16M]*

(a) Byte streams: FileInputStream/FileOutputStream with copy file example. (4M)
(b) Character streams: FileReader/FileWriter with BufferedReader/BufferedWriter. (6M)
(c) Try-with-resources syntax and reading entire file line by line. (6M)

Answer: _____

Q2. Explain generics in Java with Generic class, Generic method, and bounded types. Write programs to demonstrate a generic Stack class and a generic method to find the maximum. *[CO4 | UNIT IV | 16M]*

(a) Generic class Box<T> and Pair<K,V> with multiple type parameters. (4M)
(b) Generic method with upper bound <T extends Comparable<T>> to find max. (6M)
(c) Generic Stack<T> implementation with push, pop, isEmpty. (6M)

Answer: _____

Q3. Explain wildcards and restrictions in Java Generics. Write programs demonstrating upper-bounded, lower-bounded and unbounded wildcards. *[CO4 | UNIT IV | 16M]*

(a) Unbounded wildcard List<?> and its use case. (4M)
(b) Upper-bounded wildcard <? extends Number> with sum method. (6M)
(c) Lower-bounded wildcard <? super Integer> and 5 major generics restrictions. (6M)

Answer: _____

Q4. Explain String class methods with examples. Write a Java program to perform string operations: palindrome check, count vowels, reverse words, and frequency of a character. *[CO4 | UNIT IV | 16M]*

(a) At least 10 String class methods with syntax and examples. (6M)
(b) Palindrome check and vowel count programs. (4M)
(c) Reverse words in a sentence and character frequency count. (6M)

Answer: _____

Q5. Compare String, StringBuffer, and StringBuilder with programs. Explain all important StringBuffer methods (append, insert, delete, replace, reverse, indexOf). *[CO4 | UNIT IV | 16M]*

(a) Comparison table: mutability, thread safety, performance with memory diagram. (4M)
(b) StringBuffer methods: append(), insert(), delete(), reverse(), replace(). (6M)
(c) Performance benchmark program showing why StringBuilder is faster in loops. (6M)

Answer: _____

UNIT V
JavaFX Event Handling, Controls and Components

PART A 2-Mark Questions with Answers

Q1. What is JavaFX and what are its key components? *[CO5 | UNIT V | 2M]*

Answer:

JavaFX is Java's modern UI framework for building rich desktop applications.

Key components: Stage (top-level window), Scene (content container), Node (UI element), Scene Graph (hierarchy of nodes).

All JavaFX apps extend javafx.application.Application and override the start(Stage) method.

Q2. What is a Scene Graph in JavaFX? *[CO5 | UNIT V | 2M]*

Answer:

A Scene Graph is a hierarchical tree of nodes that represents the entire visual content of a JavaFX application.

Root node → Layout nodes (Pane, VBox, GridPane) → Leaf nodes (Button, Label, TextField).

Each node can be styled, transformed, and listen for events.

Q3. What is an event in JavaFX? *[CO5 | UNIT V | 2M]*

Answer:

An event is a notification that something has happened — typically a user interaction.

JavaFX events are represented as objects of EventType classes.

Common events: ActionEvent (button click), MouseEvent (mouse click/move), KeyEvent (key press/release).

Q4. How do you handle a button click event in JavaFX? *[CO5 | UNIT V | 2M]*

Answer:

Use setOnAction() method and provide an EventHandler or lambda expression.

Example: btn.setOnAction(e -> { System.out.println("Clicked!"); });

Or: btn.setOnAction(new EventHandler<ActionEvent>() { public void handle(ActionEvent e){ } });

Q5. Differentiate TextField and TextArea in JavaFX. *[CO5 | UNIT V | 2M]*

Answer:

TextField: single-line text input control; used for short inputs like name, email.

TextArea: multi-line text input control; used for longer text like addresses, comments.

Both extend TextInputControl. Key methods: getText(), setText(), setPromptText().

Q6. What are RadioButtons and ToggleGroup in JavaFX? *[CO5 | UNIT V | 2M]*

Answer:

RadioButton is a control that can be selected or deselected, representing a choice.
ToggleGroup ensures mutual exclusivity — only one RadioButton in a group can be selected at a time.
Example: ToggleGroup g = new ToggleGroup(); rb1.setToggleGroup(g); rb2.setToggleGroup(g);

Q7. What is the difference between ComboBox and ChoiceBox? *[CO5 | UNIT V | 2M]*

Answer:
ComboBox: shows a dropdown list; optionally editable (allows typing custom value); more feature-rich.
ChoiceBox: shows a simple dropdown; not editable; lighter weight.
Both use getItems() to add options and getValue() to get selected value.

Q8. Name four JavaFX layouts and their purpose. *[CO5 | UNIT V | 2M]*

Answer:
1. HBox – arranges nodes in a horizontal row.
2. VBox – arranges nodes in a vertical column.
3. GridPane – arranges nodes in a grid of rows and columns.
4. BorderPane – has 5 regions: Top, Bottom, Left, Right, Center.

Q9. What is a MenuBar in JavaFX? *[CO5 | UNIT V | 2M]*

Answer:
MenuBar is a horizontal bar displayed at the top of a window that contains Menu objects.
Menu → contains MenuItem objects (clickable actions).
Example: MenuBar → File Menu → Open, Save, Exit MenuItems.

Q10. What is the difference between FlowPane and GridPane? *[CO5 | UNIT V | 2M]*

Answer:
FlowPane: arranges nodes in a flow (like text wrapping); automatically wraps to next row/column when space runs out.
GridPane: arranges nodes in a strict grid with defined rows and columns; precise positioning using add(node, col, row).

PART B 16-Mark Questions (Answer Space Provided)

Q1. Explain the architecture of a JavaFX application. Write a complete JavaFX program for a Student Registration Form with Name, Age, Gender (RadioButton), Course (ComboBox), and a Submit button. *[CO5 | UNIT V | 16M]*

(a) JavaFX application structure: Application class, start(), Stage, Scene, Scene Graph. (4M)
(b) Student Registration Form using GridPane layout with all controls. (8M)
(c) Event handling: Submit button displays entered data in a Label or Alert. (4M)

Answer: _____

Q2. Explain JavaFX event handling in detail. Write programs to handle MouseEvent, KeyEvent, and ActionEvent. Demonstrate event propagation (bubbling and capturing). *[CO5 | UNIT V | 16M]*

(a) Event class hierarchy and EventHandler interface with lambda syntax. (4M)
(b) MouseEvent: onClick, onMouseMoved, coordinates display on Pane. (6M)
(c) KeyEvent: track key typed/pressed in a TextField; ActionEvent on Button. (6M)

Answer: _____

Q3. Explain all JavaFX layout managers (HBox, VBox, GridPane, BorderPane, FlowPane, StackPane, AnchorPane) with diagrams and code examples. *[CO6 | UNIT V | 16M]*

(a) HBox and VBox with spacing, padding, and alignment properties. (4M)
(b) GridPane with column/row spanning; BorderPane with all 5 regions populated. (6M)
(c) FlowPane, StackPane, and AnchorPane with examples and use cases. (6M)

Answer: _____

Q4. Explain JavaFX controls: CheckBox, ListView, ScrollPane, Slider, and ProgressBar. Write a program that uses at least four of these controls together in a meaningful UI. *[CO6 | UNIT V | 16M]*

(a) CheckBox with isSelected(), indeterminate state. (4M)
(b) ListView with MultipleSelectionModel; ScrollPane wrapping content. (6M)
(c) Slider with value change listener updating a ProgressBar in real time. (6M)

Answer: _____

Q5. Explain the JavaFX Menu system in detail. Write a complete program implementing MenuBar with File, Edit, and Help menus including CheckMenuItem, RadioMenuItem, and keyboard shortcuts (accelerators). *[CO6 | UNIT V | 16M]*

(a) MenuBar, Menu, MenuItem, SeparatorMenuItem hierarchy with code. (4M)
(b) File menu: New, Open, Save, separator, Exit with setOnAction handlers. (6M)
(c) CheckMenuItem in View menu, RadioMenuItem in Format menu, keyboard shortcuts with KeyCombination. (6M)

Answer: _____