VEL TECH HIGH TECH

Dr.RANGARAJAN Dr.SAKUNTHALA ENGINEERING COLLEGE

An Autonomous Institution

Approved by AICTE-New Delhi, Affiliated to Anna University, Chennai

Accredited by NBA, NewDelhi &Accredited by NAAC with "A" Grade &CGPAof3.27

DEPARTMENT OF INFORMATION TECHNOLOGY

INNOVATIVE ASSIGNMENT-II

DEGREE	**B.TECH(IT)**	**FACULTYNAME:S.NITHYA**	
COURSECODE:	**21IT35IT**	**COURSENAME:**	**OBJECT ORIENTED PROGRAMMING**
YEAR/SEM/SEC	**II/ III / B**	**STUDENTSSTRENGTH/ BATCHES**	**64 /20**

S. NO	STUDENT NAME	PROBLEMSTATEMENT	CO	K LEVEL
		BATCH 1		
1	PRIYAR	Design an application-specific exception classthat represents a unique error condition in your program. Demonstrate how this custom exception is thrown and caught within your application. Skeleton: • Introduction of creating own exception. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References	CO3 & CO4	K3
2	SIVANIVB	Implement a set of methods for basic string manipulations such as concatenation, substring extraction, and length calculation. Demonstrate these methods in a practical context. Skeleton: • Introduction of strings. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References		
3	VARSHAA	Design a generic event-handling system. Create generic event classes and a mechanism for handling events in a flexible and type-safe manner. Skeleton: • Introduction of generic programming. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References		

S. NO	STUDENT NAME	PROBLEM STATEMENT	CO	K LEVEL
		BATCH 2		
4	SUREKHAM	Implement a class that supports basic object cloning using the Cloneable interface. Demonstrate how cloning works for simple objects and discuss its limitations. Skeleton: • Introduction of object cloning. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References	CO3 & CO4	K3
5	TONIYAAG J	Create a generic container class that can hold objects of any type but is upper-bounded to accept only objects that implement a specific interface. Demonstrate how this constraint enhances the container's functionality. Skeleton: • Introduction of bounded types. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References		
6	YAZHINI S	Create an inner class that is used for encapsulation. The inner class should have access to private members of the outer class, and demonstrate scenarios where this encapsulation is beneficial. Skeleton: • Introduction of inner class. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References		

S. NO	STUDENT NAME	PROBLEM STATEMENT	CO	K LEVEL
		BATCH 3		
7	PRINCY VINOFER V	Design a situation where your program can gracefully degrade its functionality in the face of exceptions. Implement recovery mechanisms to allow the program to continue operating in a degraded state. **Skeleton:** • Introduction of throwing and catching exception. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References	CO3 & CO4	K3
8	THEJASHREEN	Create a utility class that provides custom string manipulation methods, such as reversing a string, removing duplicate characters, or converting to uppercase or lowercase selectively. **Skeleton:** • Introduction of strings. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References		
9	PUSHPALATHA E	Implement a scenario where an enum uses generics. Demonstrate how this approach can be useful when the enum needs to associate with different types of values. **Skeleton:** • Introduction of generic programming. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References		

S. NO	STUDENT NAME	PROBLEM STATEMENT	CO	K LEVEL
		BATCH 4		
10	SOUNDARIYAN	Address considerations related to object cloning in a multithreaded environment. Implement synchronization mechanisms or other strategies to ensure safe cloning in concurrent scenarios. **Skeleton:** • Introduction of object cloning. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References	CO3 & CO4	K3
11	SANTHIYAG	Implement a method that accepts a list of elements and a lower-bounded wildcard. The method should add elements to the list only if they are of the type specified by the lower-bounded wildcard. **Skeleton:** • Introduction of bounded types. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References		
12	SREEDEVIS	Implement a local inner class inside a method. Demonstrate how local inner classes can provide a way to structure code logically within a specific scope and discuss situations where this is advantageous. **Skeleton:** • Introduction of inner class. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References		

S. NO	STUDENT NAME	PROBLEM STATEMENT	CO	K LEVEL
		BATCH 5		
13	VIJAYALAKSHMI C	Extend a custom exception to support internationalization (i18n). Implement a mechanism to display error messages in different languages based on the user's locale. **Skeleton:** • Introduction of creating own exception. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References	CO3 & CO4	K3
14	MONIKAS	Develop scenarios that involve comparing strings using different methods (e.g., equals, compareTo). Implement a string search mechanism to find occurrences of a specific substring. **Skeleton:** • Introduction of strings. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References		
15	YOGASREES	Develop a program that involves user input. Implement a robust input validation mechanism that throws and catches exceptions when invalid input is provided. Provide informative error messages to the user. **Skeleton:** • Introduction of built in exception • Explain with algorithm and program. • Outputscreenshot, Conclusionand References		

S. NO	STUDENT NAME	PROBLEM STATEMENT	CO	K LEVEL
		BATCH 6		
16	NOVENJOVI J	Extend your cloning mechanism to support deep copying. Implement a scenario where the object being cloned contains nested objects, and ensure that all nested objects are also cloned. **Skeleton:** • Introduction of object cloning. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References	CO3 & CO4	K3
17	NAVEEN KUMAR M	Design a generic sorting algorithm that works with a list of elements but is constrained to elements that extend a particular class and implement a specific interface. Demonstrate the dual-bounded sorting in action. **Skeleton:** • Introduction of bounded types. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References		
18	MOHAMED NAZURUDEEN K	Create a scenario where an inner class implements an interface defined in the outer class. Demonstrate how this relationship enhances encapsulation and code organization. **Skeleton:** • Introduction of inner class. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References		

S. NO	STUDENT NAME	PROBLEM STATEMENT	CO	K LEVEL
		BATCH 7		
19	SHIJI GOVINP	Create a scenario involving multithreading where a custom exception needs to be thrown and handled. Discuss the challenges and considerations related to concurrent exception handling. **Skeleton:** • Introduction of creating own exception. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References	CO3 & CO4	K3
20	RAJKUMARP	Explore the use of regular expressions for advanced string matching. Implement a scenario where regular expressions are used to validate and extract information from strings. **Skeleton:** • Introduction of strings. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References		
21	SATHISH H	Create a scenario where an exception triggers another exception (chained exceptions). Implement a mechanism to log and handle these chained exceptions effectively. **Skeleton:** • Introduction of built in exception • Explain with algorithm and program. • Outputscreenshot, Conclusionand References		
22	PRIYANRAJ I P	Design a program that involves network or file I/O operations. Explore limitations related to network latency or slow disk I/O, and implement strategies to enhance performance or gracefully handle delays. **Skeleton:** • Introduction of bounded types restrictions and limitation • Explain with algorithm and program. • Outputscreenshot, Conclusionand References		

S. NO	STUDENT NAME	PROBLEM STATEMENT	CO	K LEVEL
		BATCH 8		
23	SAKTHIVELR	Design and implement a custom cloning interface that defines a method for cloning objects. Use this interface to enable cloning for classes that don't implement the standard Cloneable interface. **Skeleton:** • Introduction of object cloning. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References	CO3 &CO4	K3
24	RUBAN PRASATH R	Create a generic method that accepts two parameters with different upper bounds. Demonstrate how the method can be used with various combinations of types, ensuring that the bounds are respected. **Skeleton:** • Introduction of bounded types. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References		
25	SURESH A	Design a scenario involving inheritance with inner classes. Explore situations where inner classes in subclasses interact with inner classes in the superclass. **Skeleton:** • Introduction of inner class. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References		

S. NO	STUDENT NAME	PROBLEM STATEMENT	CO	K LEVEL
		BATCH 9		
26	SAKTHINAGA SARVESH A	Integrate your custom exceptions with a logging system. Demonstrate how exception information can be logged at different severity levels and include relevant details to aid in debugging. **Skeleton:** • Introduction of creating own exception. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References	CO3 & CO4	K3
27	VASANTH S	Implement a program that showcases different string formatting options. Use techniques like String.format and demonstrate how they can improve the presentation of textual information. **Skeleton:** • Introduction of strings. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References		
28	SANTHOSH T	Implement file I/O operations (reading and writing to files). Demonstrate how exceptions related to file operations (e.g., IOException) can be caught and handled gracefully. **Skeleton:** • Introduction of built in exception • Explain with algorithm and program. • Outputscreenshot, Conclusionand References		

S. NO	STUDENT NAME	PROBLEM STATEMENT	CO	K LEVEL
		BATCH 10		
29	SHADISH.G	Implement a scenario where object cloning is used in the context of the Prototype design pattern. Demonstrate how cloning can be employed to create new instances based on existing prototypes. **Skeleton:** • Introduction of object cloning. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References	CO3 & CO4	K3
30	VISHWAS	Design a scenario where bounded generics play a crucial role in inheritance. Create a base class with a bounded generic type, and extend it with subclasses that specialize the generic type further. **Skeleton:** • Introduction of bounded types. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References		
31	NITHISH KUMAR D	Explore limitations associated with garbage collection, such as pause times or limitations in fine-tuning. Implement a program that generates heavy garbage collection loads and discuss strategies for minimizing their impact. **Skeleton:** • Introduction of bounded types restrictions and limitation • Explain with algorithm and program. • Outputscreenshot, Conclusionand References		

S. NO	STUDENT NAME	PROBLEM STATEMENT	CO	K LEVEL
		BATCH 11		
32	MONISHN	Design a simplified framework where users can define their own custom exceptions to handle specific situations within the framework. Demonstrate how this approach enhances extensibility. **Skeleton:** • Introduction of creating own exception. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References	CO3 & CO4	K3
33	SHAIK ROSHANAHMAD	Compare the performance of StringBuilder and StringBuffer in scenarios involving string concatenation. Design a program that measures and analyzes the execution time of both approaches. **Skeleton:** • Introduction of strings. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References		
34	MOHAMMEDNOWFALRAIZI	Investigate limitations related to using primitive types with generics. Demonstrate scenarios where the lack of primitive type support in generics impacts performance or code readability. **Skeleton:** • Introduction of bounded types restrictions and limitation • Explain with algorithm and program. • Outputscreenshot, Conclusionand References		

S. NO	STUDENT NAME	PROBLEM STATEMENT	CO	K LEVEL
		BATCH 12		
35	RAJASEDHUPATHIG	Explore how to handle cloning for immutable objects. Implement a scenario where an object is immutable, and demonstrate how cloning can be adapted to work in such cases. **Skeleton:** • Introduction of object cloning. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References	CO3 & CO4	K3
36	SARANMANIM	Design a generic stack class with a method that accepts another stack with a lower-bounded wildcard. The method should pop elements from the input stack and push them onto the current stack. **Skeleton:** • Introduction of bounded types. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References		
37	MANIGANDANG	Implement a scenario involving object serialization. Demonstrate limitations related to serializing complex objects or objects with non-serializable components, and propose solutions. **Skeleton:** • Introduction of generic methods. • Explain with algorithm and program. • Outputscreenshot, Conclusion and References		

S.NO	STUDENT NAME	PROBLEM STATEMENT	CO	K LEVEL
		BATCH 13		
38	MOHAMED ARIF AKTHAR A	Implement a custom exception to handle validation errors within your application. Show how this exception can be used to communicate specific validation failures to the calling code. **Skeleton:** • Introduction of creating own exception. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References	CO3 & CO4	K3
39	SANJAY N	Discuss the immutability of strings in Java and implement scenarios where creating new strings might lead to memory inefficiencies. Explore strategies to optimize memory usage. **Skeleton:** • Introduction of strings. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References		
40	VISHAL JAYSURYA M	Design a scenario where memory limitations become a critical factor. Implement strategies to manage and optimize memory usage, considering scenarios like large data sets or constrained environments. **Skeleton:** • Introduction of generic methods. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References		
41	YOGESHWARAN R	Experiment with different garbage collection tuning parameters (e.g., heap size, garbage collector type). Implement a program that allows users to adjust these parameters and observe their impact on garbage collection behavior. **Skeleton:** • Introduction of garbage collection. • Explain with algorithm and program. • Output screenshot, Conclusion and References		

S. NO	STUDENT NAME	PROBLEM STATEMENT	CO	K LEVEL
		BATCH 14		
42	PRAKASHM	Introduce annotations to your classes to mark fields that should or should not be cloned. Implement a cloning mechanism that respects these annotations, allowing selective cloning. **Skeleton:** • Introduction of object cloning. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References	CO3 & CO4	K3
43	SIVABALANT	Design a multithreaded program that utilizes garbage collection. Discuss challenges and considerations related to garbage collection in a concurrent context, and implement strategies to address them. **Skeleton:** • Introduction of garbage collection. • Explain with algorithm and program. • Outputscreenshot, Conclusion and References		
44	TAMILSELVAN V	Implement a program that demonstrates the use of bulk operations provided by the addAll and removeAll methods. Explore how these operations can simplify working with multiple ArrayLists. **Skeleton:** • Introduction of array list. • Explain with algorithm and program. • Outputscreenshot, Conclusion and References		

S. NO	STUDENT NAME	PROBLEM STATEMENT	CO	K LEVEL
		BATCH 15		
45	RAHULSRIS	Develop a custom exception that is aware of the context in which it is thrown. Illustrate how this information can be leveraged to provide more meaningful error messages or to guide recovery processes. Skeleton: • Introduction of creating own exception. • Explain with algorithm and program. • Output screenshot, Conclusion and References	CO3 & CO4	K3
46	ROHANUBENDRAM	Implement a program that uses string interning and pooling. Demonstrate scenarios where string pooling is beneficial and discuss considerations when deciding to use string interning. **Skeleton:** • Introduction of strings. • Explain with algorithm and program • Output screenshot, Conclusion and References		
47	SARANS	Adapt your program to simulate a resource-constrained environment, such as an IoT device. Implement and discuss garbage collection strategies suitable for devices with limited memory resources. **Skeleton:** • Introduction of garbage collection. • Explain with algorithm and program. • Outputscreenshot, Conclusion and References		

S. NO	STUDENT NAME	PROBLEM STATEMENT	CO	K LEVEL
		BATCH 16		
48	ROHITH G	Explore how to handle cloning for immutable objects. Implement a scenario where an object is immutable, and demonstrate how cloning can be adapted to work in such cases. **Skeleton:** • Introduction of object cloning. • Explain with algorithm and program. • Output screenshot, Conclusion and References	CO3 & CO4	K3
49	SANJAY S	Create a class representing custom objects and populate an ArrayList with instances of this class. Demonstrate how to perform operations such as sorting or searching on an ArrayList of custom objects. **Skeleton:** • Introduction of array list. • Explain with algorithm and program. • Outputscreenshot, Conclusion and References		
50	SANTHOSH V	Explore real-time garbage collection scenarios. Implement a program where garbage collection needs to be performed with minimal impact on the application's responsiveness. Discuss challenges and strategies for achieving real-time collection. **Skeleton:** • Introduction of garbage collection. • Explain with algorithm and program. • Outputscreenshot, Conclusion and References		

S. NO	STUDENT NAME	PROBLEM STATEMENT	CO	K LEVEL
		BATCH 17		
51	PRAVEEN V	Create a scenario where multiple exceptions can occur in different layers of your application. Implement exception chaining to provide a comprehensive view of the problem when an error occurs. **Skeleton:** • Introduction of creating own exception. • Explain with algorithm and program. • Output screenshot, Conclusion and References	CO3 & CO4	K3
52	SAMEER UR RAHMAN	Explore the concept of internationalization (i18n) and localization (l10n) with strings. Implement a program that displays messages in different languages based on the user's locale. **Skeleton:** • Introduction of strings. • Explain with algorithm and program. • Outputscreenshot, Conclusion and References		
53	SUBASHK	Design a scenario where performance is critical. Implement a program that measures the time taken for various ArrayList operations (e.g., add, remove, search) and explores strategies for optimizing performance. **Skeleton:** • Introduction of array list. • Explain with algorithm and program. • Output screenshot, Conclusion and References		
54	MONISHKUMAR N	Design a set of custom unchecked exceptions that capture unexpected errors or exceptional conditions in your application. Demonstrate how these exceptions are thrown and caught. **Skeleton:** • Introduction of throwing and catching exception. • Explain with algorithm and program. • Outputscreenshot, Conclusion and References		

S. NO	STUDENT NAME	PROBLEM STATEMENT	CO	K LEVEL
		BATCH 18		
55	SHREE SHAINIHA J S	Extend a custom exception to include additional information (e.g., error codes, timestamps, or user-specific details). Illustrate how this extra information can be useful during exception handling. **Skeleton:** • Introduction of creating own exception. • Explain with algorithm and program. • Outputscreenshot, Conclusion and References	CO3 & CO4	K3
56	TANUSHRI S	Extend your custom exception hierarchy to allow user-defined exception messages. Implement scenarios where informative error messages are provided to users, helping them understand the nature of the exceptions. **Skeleton:** • Introduction of exception hierarchy. • Explain with algorithm and program. • Outputscreenshot, Conclusion and References		
57	VARINI LAKSHMI S	Explore the use of wildcards in generic classes. Implement a generic class that uses wildcards to define flexible relationships between different generic types. **Skeleton:** • Introduction of generic class. • Explain with algorithm and program. • Outputscreenshot, Conclusion and References		

S. NO	STUDENT NAME	PROBLEM STATEMENT	CO	K LEVEL
		BATCH 19		
58	BALAKRISHNAN N	Implement a checked custom exception that encapsulates a specific business rule violation in your application. Demonstrate how this exception is thrown and caught within the context of a user interaction. **Skeleton:** • Introduction of creating own exception. • Explain with algorithm and program. • Output screenshot, Conclusion and References	CO3 & CO4	K3
59	MANOJKUMAR G	Implement a program where exceptions cascade through different layers of your application. Demonstrate how exceptions at higher levels can catch and wrap exceptions at lower levels, providing a more meaningful context. **Skeleton:** • Introduction of exception hierarchy. • Explain with algorithm and program. • Output screenshot, Conclusion and References		
60	VIDHYAPATHY. M	Implement a program that dynamically instantiates generic classes based on user input. Demonstrate how reflection can be used to work with generic classes dynamically. **Skeleton:** • Introduction of generic class. • Explain with algorithm and program. • Output screenshot, Conclusion and References		

S. NO	STUDENT NAME	PROBLEM STATEMENT	CO	K LEVEL
		BATCH 20		
61	JOHN BRITOW AROKKIAR RAJ R	Create a hierarchy of custom exceptions that reflects different levels of severity or specificity in error conditions. Illustrate scenarios where each type of exception would be appropriate. **Skeleton:** • Introduction of creating own exception. • Explain with algorithm and program. • Output screenshot, Conclusion and References	CO3 & CO4	K3
62	PRAKADESHG	Integrate a visual representation of your custom exception hierarchy into your program. Provide a dynamic visualization that allows users to explore the relationships between different exception types. **Skeleton:** • Introduction of exception hierarchy. • Explain with algorithm and program. • Output screenshot, Conclusion and References		
63	KIRTHICK KUMAR R	Implement a generic class with bounds. Demonstrate scenarios where bounds on the generic type are beneficial, and discuss the advantages of constraining the types that can be used. **Skeleton:** • Introduction of generic class. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References		
64	SUNIL KUMAR	Create a hierarchy of custom checked exceptions that represent various business rule violations in your application domain. Implement scenarios where these exceptions are thrown and caught. **Skeleton:** • Introduction of throwing and catching exception. • Explain with algorithm and program. • Outputscreenshot, Conclusionand References		

HOD **DEAN ACEDEMICS**