

UNIT IV I/O, GENERICS, STRING HANDLING

S.NO	PART -A				
1.	Define Java I/O basics. Java I/O (Input and Output) is used to process the input and produce the output. Java uses the concept of stream to make I/O operation fast. These streams support all the types of objects, data-types, characters, files etc to fully execute the I/O operations. The java.io package contains all the classes required for input and output operations  <pre> graph LR Source[Source] -- "Inputstream READ" --> JavaApp[JAVA APPLICATION] JavaApp -- "Outputstream Write" --> Destination[Destination] </pre>				
2.	What are Streams? A Stream is a sequence of data or it is an abstraction that either produces or consumes information. In other simple words it is a flow of data from which you can read or write data to it. It's called a stream because it's like a stream of water that continues to flow.				
3.	What are the Predefined streams? In java, 3 streams are created for us automatically. All these streams are attached with console. 1. System.in 2. System.out 3. System.err				
4.	Mention the types of streams. 1. Byte Stream – Byte Streams provide a convenient means of handling input and output in terms of bytes. Byte streams are used when reading or writing binary data. 2. Character Stream – Character streams provide a convenient means of handling input or output in terms of characters. In some cases, character streams are more efficient than byte streams.				
5.	Define Print Stream. The PrintStream class provides methods to write data to another stream. The PrintStream class automatically flushes the data so there is no need to call flush() method. Moreover, its methods don't throw IOException.				
6.	Mention the important methods of Buffered Reader and Writer class. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%;">Important methods of BufferedWriter Class</th><th style="width: 50%;">Important methods of BufferedReader Class</th></tr> </thead> <tbody> <tr> <td>void write(int ch) throws IOException void write(String s) throws IOException void write(char[] ch) throws IOException void newLine() for inserting a new line character. void flush() void close()</td><td>int read() int read(char [] ch) String readLine(); - Reads the next line present in the file. If there is no nextline this method returns null. void close()</td></tr> </tbody> </table>	Important methods of BufferedWriter Class	Important methods of BufferedReader Class	void write(int ch) throws IOException void write(String s) throws IOException void write(char[] ch) throws IOException void newLine() for inserting a new line character. void flush() void close()	int read() int read(char [] ch) String readLine(); - Reads the next line present in the file. If there is no nextline this method returns null. void close()
Important methods of BufferedWriter Class	Important methods of BufferedReader Class				
void write(int ch) throws IOException void write(String s) throws IOException void write(char[] ch) throws IOException void newLine() for inserting a new line character. void flush() void close()	int read() int read(char [] ch) String readLine(); - Reads the next line present in the file. If there is no nextline this method returns null. void close()				
7.	Define Print Writer Class. The Java.io.PrintWriter class prints formatted representations of objects to a text- output stream. PrintWriter defines several constructors.				

	PrintWriter(OutputStream outputStream, boolean flushOnNewline) PrintWriter supports the print() and println() methods for all types including Object. Thus, you can use these methods in the same way as they have been used with System.out. If an argument is not a simple type, the PrintWriter methods call the object's toString() method and then print the result. To write to the console by using a PrintWriter, specify System.out for the output stream and flush the stream after each newline. For example, this line of code creates a PrintWriter that is connected to console output: PrintWriter pw = new PrintWriter(System.out, true);
8.	What are the 3 ways to read input from console in Java? 1. Using Buffered Reader Class 2. Using Scanner Class 3. Using Console Class
9.	Define Java Console Class. The Java Console class is be used to get input from console. It provides methods to read texts and passwords. If you read password using Console class, it will not be displayed to the user. The java.io.Console class is attached with system console internally. 1. String text=System.console().readLine(); 2. System.out.println("Text is: "+text);
10.	What is File Handling in Java? File handling in Java implies reading from and writing data to a file. The File class from the java.io package, allows us to work with different formats of files. In order to use the File class, you need to create an object of the class and specify the filename or directory name.
11.	What are the File Operations in Java? 1) Create a File 2) Get File Information 3) Write To a File 4) Read from a File
12.	What is Generic Programming? Generic programming is a style of computer programming in which algorithms are written in terms of “to-be-specified-later” types that are then instantiated when needed for specific types provided as parameters. Generic programming refers to writing code that will work for many types of data.
13.	What is non-generic and its limitations? In java, there is an ability to create generalized classes, interfaces and methods by operating through Object class. Limitation of non-generic: 1) Explicit casts must be employed to retrieve the stored data. 2) Type mismatch errors cannot be found until run time.
14.	What is the need for Generic Programming? Need for Generic: 1. It saves the programmers burden of creating separate methods for handling data belonging to different data types. 2. It allows the code reusability. 3. Compact code can be created.
15.	What are the Advantages of Java Generics? 1) Code Reuse 2) Type-safety 3) Elimination of casts 4) Stronger type checks at compile time 5) Enabling programmers to implement generic algorithms.

17.	What are Generic Methods? A Generic Method is a method with type parameter. We can write a single generic method declaration that can be called with arguments of different types. Based on the types of the arguments passed to the generic method, the compiler handles each method call appropriately. <pre>Syntax modifiers <TypeVariable₁, TypeVariable₂, . . .> returnType methodName(parameters) { body }</pre>												
18.	What are the rules to define Generic methods? • All generic method declarations have a type parameter section delimited by angle brackets (< and >) that precedes the method's return type. • Each type parameter section contains one or more type parameters separated by commas. A type parameter, also known as a type variable, is an identifier that specifies a generic type name. • The type parameters can be used to declare the return type and act as placeholders for the types of the arguments passed to the generic method, which are known as actual type arguments. • A generic method's body is declared like that of any other method. Note that type parameters can represent only reference types, not primitive types (like int, double and char).												
19.	What are Bounded Type Parameters? Bounded Type Parameter is a type parameter with one or more bounds. The bounds restrict the set of types that can be used as type arguments and give access to the methods defined by the bounds. Syntax: <T extends superclass>												
20.	What are wild card arguments? Question mark (?) is the wildcard in generics and represents an unknown type. The wildcard can be used as the type of a parameter, field, or local variable and sometimes as a return type. <table><thead><tr><th>Name</th><th>Syntax</th><th>Meaning</th></tr></thead><tbody><tr><td>Wildcard with lower bound</td><td>? extends B</td><td>Any subtype of B</td></tr><tr><td>Wildcard with upper bound</td><td>? super B</td><td>Any supertype of B</td></tr><tr><td>Unbounded wildcard</td><td>?</td><td>Any type</td></tr></tbody></table>	Name	Syntax	Meaning	Wildcard with lower bound	? extends B	Any subtype of B	Wildcard with upper bound	? super B	Any supertype of B	Unbounded wildcard	?	Any type
Name	Syntax	Meaning											
Wildcard with lower bound	? extends B	Any subtype of B											
Wildcard with upper bound	? super B	Any supertype of B											
Unbounded wildcard	?	Any type											
21.	What is Bounded and Unbounded Wild Card? BOUNDED WILDCARDS: - A bounded wildcard is a wildcard with either an upper or a lower bound. UNBOUNDED WILDCARD: - Unbounded wildcard can be used where generic method to be working with all types. In this case the wildcard “?” simply matches any valid objects.												
22.	Define Strings. String is a sequence of characters. But in Java, a string is an object that represents a sequence of characters. The java.lang.String class is used to create string object.												
23.	How to create String object? There are two ways to create a String object: 1. By string literal Example: String s=“Welcome”; 2. By new keyword Example: String s=new String(“Welcome”);												

24.	Define Java String Pool. Java String pool refers to collection of Strings which are stored in heap memory. In this, whenever a new object is created, 1) String pool first checks whether the object is already present in the pool or not. 2) If it is present, then same reference is returned to the variable 3) else new object will be created in the String pool and the respective reference will be returned.													
25.	What is String Comparison in Java? We can compare two given strings on the basis of content and reference. It is used in authentication (by equals() method), - sorting (by compareTo() method), - reference matching (by == operator) etc.													
26.	What are the ways to compare string objects? There are three ways to compare String objects: 1. By equals() method 2. By == operator 3. By compareTo() method													
27.	Differentiate String class and StringBuffer class <table border="1"> <thead> <tr> <th>S.No</th><th>String Class</th><th>StringBuffer Class</th></tr> </thead> <tbody> <tr> <td>1</td><td>String objects are constants and Immutable (cannot change the content)</td><td>StringBuffer objects are not constants and mutable (can change the content)</td></tr> <tr> <td>2</td><td>String class supports constant strings</td><td>StringBuffer class supports growable and modifiable strings</td></tr> <tr> <td>3</td><td>The methods in the String class are not synchronized</td><td>The methods of the StringBuffer class can be synchronized</td></tr> </tbody> </table>		S.No	String Class	StringBuffer Class	1	String objects are constants and Immutable (cannot change the content)	StringBuffer objects are not constants and mutable (can change the content)	2	String class supports constant strings	StringBuffer class supports growable and modifiable strings	3	The methods in the String class are not synchronized	The methods of the StringBuffer class can be synchronized
S.No	String Class	StringBuffer Class												
1	String objects are constants and Immutable (cannot change the content)	StringBuffer objects are not constants and mutable (can change the content)												
2	String class supports constant strings	StringBuffer class supports growable and modifiable strings												
3	The methods in the String class are not synchronized	The methods of the StringBuffer class can be synchronized												
28.	What is absolute file name and relative file name The filename that can be specified with the complete path name and drive letter is called absolute file name. The file name which is specified as a path relative to the current directory is called relative file name. For example new File("myprogram.html");													
29	Write a Java code to check if the command line argument is file or not <pre> class Test { public static void main(String[] args) { File obj=new File(args[0]); If(obj.isFile()) { System.out.println("This is a file name" + obj.getPath()); } } } </pre>													