Internal Assessment Test 3 March 2024									
Sub:	**Object Oriented Programming with Java**				Sub Code:	BCS306A	Branch:	AIML/CSE(AIML)	
Date:	4 -03-24	Duration:90 m	Max Marks:	50	Sem /Sec:	III A/B/C		OBE	

	Answer any FIVE FULL Questions	Marks	CO	RBT
1 (a)	What is Thread? Explain the two ways of creating a Thread in Java	10	CO5	L2

Thread: A multithreaded program contains two or more parts that can run concurrently. Each such part of a program is called **thread**.

Threads are lightweight tasks:

1. they share the same address space of the process they belong to
2. they cooperatively share the same process
3. inter-thread communication is inexpensive
4. context-switching from one thread to another is low-cost

Explain two ways of creating a thread in JAVA with example.

Following are the two ways of creating a new thread:

1. by implementing the Runnable interface
2. by extending the Thread class

1. To create a new thread by implementing the Runnable interface:

2. create a class that implements the run method (inside this method, we define the code that constitutes the new thread):

public void run()

3. Instantiate a Thread object within that class, a possible constructor is:
   a. **Thread(Runnable threadOb, String threadName)**

4. Call the start method on this object (start calls run

void start()

```
NewThread() {
// Create a new, second thread
t = new Thread(this, "Demo Thread");
System.out.println("Child thread: " + t);
t.start(); // Start the thread
}
// This is the entry point for the second thread.
public void run() {
try {
for(int i = 5; i > 0; i--) {
System.out.println("Child Thread: " + i);
Thread.sleep(500);
}
} catch (InterruptedException e) {
System.out.println("Child interrupted.");
```

```
}
System.out.println("Exiting child thread.");
}
}
class ThreadDemo {
public static void main(String args[ ] ) {
new NewThread(); // create a new thread
try {
for(int i = 5; i > 0; i--) {
System.out.println("Main Thread: " + i);
Thread.sleep(1000);
}
} catch (InterruptedException e) {
System.out.println("Main thread interrupted.");
}
System.out.println("Main thread exiting.");
}
}
```

2. The second way to create a new thread by extending Thread class

a. Create a new class that extends Thread
b. Create an instance of that class
c. Thread provides both run and start methods:
d. The extending class must override run method
e. It must also call the start method

```
class NewThread extends Thread {
NewThread() {
// Create a new, second thread
super("Demo Thread");
System.out.println("Child thread: " + this);
start(); // Start the thread
}
// This is the entry point for the second thread.
public void run() {
try {
for(int i = 5; i > 0; i--) {
System.out.println("Child Thread: " + i);
Thread.sleep(500);
}
} catch (InterruptedException e) {
System.out.println("Child interrupted.");
}
System.out.println("Exiting child thread.");
}
}
class ExtendThread {
public static void main(String args[]) {
new NewThread(); // create a new thread
try {
for(int i = 5; i > 0; i--) {
System.out.println("Main Thread: " + i);
Thread.sleep(1000);
}
} catch (InterruptedException e) {
System.out.println("Main thread interrupted.");
}
System.out.println("Main thread exiting.");
}
}
```

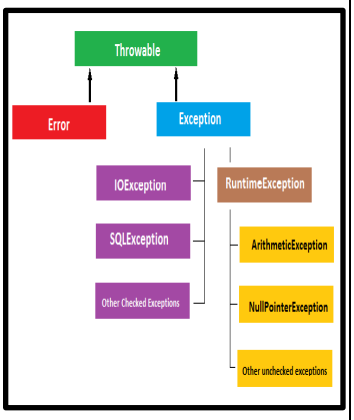
<table>
<tr>
<td>2.a</td>
<td>Explain the inbuild methods in Thread Class with an example.

<table>
<tr><th>Method</th><th>Meaning</th></tr>
<tr><td>getName</td><td>Obtain a thread's name.</td></tr>
<tr><td>getPriority</td><td>Obtain a thread's priority.</td></tr>
<tr><td>isAlive</td><td>Determine if a thread is still running.</td></tr>
<tr><td>join</td><td>Wait for a thread to terminate.</td></tr>
<tr><td>run</td><td>Entry point for the thread.</td></tr>
<tr><td>sleep</td><td>Suspend a thread for a period of time.</td></tr>
<tr><td>start</td><td>Start a thread by calling its run method.</td></tr>
</table>

// Controlling the main Thread.
class CurrentThreadDemo {
public static void main(String args[]) {
Thread t = Thread.currentThread();
System.out.println("Current thread: " + t);
// change the name of the thread
t.setName("My Thread");
System.out.println("After name change: " + t);
try {
for(int n = 5; n > 0; n--) {
System.out.println(n);
Thread.sleep(1000);
}
} catch (InterruptedException e) {
System.out.println("Main thread interrupted");
}
}
}</td>
<td>10</td>
<td>CO5</td>
<td>L2</td>
</tr>
<tr>
<td>3a</td>
<td>Explain run(),start(),methods of thread with an example

The second way to create a thread is to create a new class that extends Thread, and then to
create an instance of that class. The extending class must override the run( ) method, which
is the entry point for the new thread. It must also call start( ) to begin execution of the new
thread. Here is the preceding program rewritten to extend Thread:
// Create a second thread by extending Thread
class NewThread extends Thread {
NewThread() {
// Create a new, second thread
super("Demo Thread");
System.out.println("Child thread: " + this);
start(); // Start the thread
}
// This is the entry point for the second thread.</td>
<td>05</td>
<td>CO4</td>
<td>L3</td>
</tr>
</table>

```
public void run() {
try {
for(int i = 5; i > 0; i--) {
System.out.println("Child Thread: " + i);
Thread.sleep(500);
}
} catch (InterruptedException e) {
System.out.println("Child interrupted.");
}
System.out.println("Exiting child thread.");
}
}
class ExtendThread {
public static void main(String args[]) {
new NewThread(); // create a new thread
try {
for(int i = 5; i > 0; i--) {
System.out.println("Main Thread: " + i);
Thread.sleep(1000);
}
} catch (InterruptedException e) {
System.out.println("Main thread interrupted.");
}
System.out.println("Main thread exiting.");
}
}
```

3b.	**Explain the different type of Exception in Java.** • *Every Exception type is basically an object belonging to class* ***Exception*** • ***Throwable*** *class is the root class of Exceptions.* • ***Throwable*** *class has two direct subclasses named* ***Exception, Error*** **Checked Exceptions** • All Exceptions that extends the Exception or any one its Subclass except RunTimeException class are checked exceptions. • Checked Exceptions are checked by the Java compiler. • There are two approaches that a user can follow to deal with checked exceptions. • Inform the compiler that a method can throw an Exception. • Catch the checked exception in try catch block. • If Checked exception is caught then exception handling code will be executed and program's execution continues. • If Checked exception is not caught then java interpreter will provide the default handler. But in this case execution of the program will be stopped by displaying the name of the exceptions object. **Unchecked Exceptions** • All Exceptions that extend the RuntimeException or any one of its		5	CO4	L3

	subclass are unchecked exceptions. • Unchecked Exceptions are unchecked by compiler. • Whether you catch the exception or not compiler will pass the compilation process. • If Unchecked exception is caught then exception handling code will be executed and program's execution continues. • If Unchecked exception is not caught then java interpreter will provide the default handler. But in this case execution of the program will be stopped by displaying the name of the exceptions object.			

4. a	**What is an exception? Write the syntax for all the keywords used in Exception.**
	● **A Java exception is an object** that describes an exceptional (error) condition that has occurred in a piece of code.
	● When an exceptional condition arises, an object representing that exception is created and thrown in the method that caused the error.
	● Java exception handling is managed via five keywords:
	◦ try
	◦ catch,
	◦ throw
	◦ throws
	◦ finally
	● **Program statements that you want to monitor for exceptions are contained within a try block.**
	● If an exception occurs within the **try** block, it is thrown.
	● Your code can catch this exception, using **catch**, and handle it.
	● System-generated exceptions are automatically thrown by the Java run-time system.
	● To manually throw an exception, use the keyword **throw**.
	● Any exception that is thrown out of a method must be specified as such by **throws** clause.
	● Any code that absolutely must be executed after a try block completes is put in a **finally** block
	
	This is the general form of an exception-handling block:
	try {
	// block of code to monitor for errors
	}
	catch (ExceptionType1 exOb) {
	// exception handler for ExceptionType1
	}
	catch (ExceptionType2 exOb) {
	// exception handler for ExceptionType2
	}
	// ...
	finally {
	// block of code to be executed after try block ends
	}
	
	The general form of throw is shown here:
	throw ThrowableInstance
		10	CO4	L2

	This is the general form of a method declaration that includes a throws clause:
	type method-name(parameter-list) throws exception-list
	{
	// body of method
	}			

5a	**What are the packages and how to import the packages?** **Packages** are containers for classes that are used to keep the class namespace compartmentalized. **Packages are stored in a hierarchical manner and are explicitly imported into new class definitions.** 1. **Purpose of Importing Packages:** - In Java, all built-in classes are stored in packages. - The import statement is used to bring certain classes or entire packages into visibility, making it easier to refer to them directly without typing their full package names each time. 2. **Syntax of Import Statement:** - Import statements occur after the package statement and before any class definitions. - General form: `import pkg1[.pkg2].(classname|*);` - `pkg1` is the top-level package, `pkg2` is a subordinate package, and `classname` is the specific class to import. - `*` imports the entire package. 3. **Caution with Star Form:** - Using `*` to import entire packages may increase compilation time, especially for large packages. - It's advisable to explicitly name the classes you need to use instead of importing whole packages. 4. **Implicit Import of java.lang:** - Java implicitly imports `java.lang.*` for all programs, as many essential functions reside here. - No need to explicitly import `java.lang` classes.	4	CO4	L2

5. **Handling Conflicts:**

- If a class with the same name exists in two imported packages, the compiler stays silent.

- Compile-time error occurs if you try to use one of the classes; then, you must explicitly specify the class with its package.

6. **Optional Nature of Import Statement:**

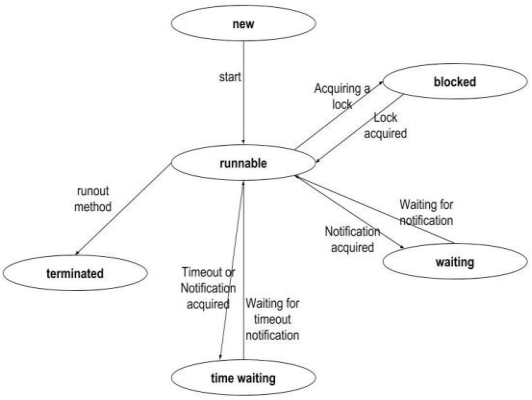
- Import statement is optional; you can use fully qualified class names instead.

- Example: `import java.util.*;` vs. `class MyDate extends java.util.Date {}`

7. **Visibility of Imported Items:**

- When a package is imported, only public items within that package are available to non-subclasses in the importing code.

- To make a class available for general use outside its package, declare it as public and put it in its own file.

5b	**Explain the states of threads in java** A thread liesonly in one of the shown states at any instant: 1) New 2) Runnable 3) Blocked 4) Waiting 5) Timed Waiting 6) Terminated  ***1.*** New Thread • When a new thread is created, it is in the new state. • The thread has not yet started to run when thread is in this state. • When a thread lies in the new state, it's code is yet to be run and hasn't started toexecute. ***2.*** *Runnable State:* • A thread that is ready to run is moved to runnable state.		CO5	L2

- In this state, a thread might actually be running or it might be ready run at any instant
  of time.

3. Blocked/Waiting state:

When a thread is temporarily inactive, then it's in one of the following states:

- Blocked
- Waiting

- For example, when a thread is waiting for I/O to complete, it lies in the blocked state.It's the responsibility of the thread scheduler to reactivate and schedule a blocked/ waiting thread.

4. Timed Waiting:

- A thread lies in timed waiting state when it calls a method with a time out parameter.
- A thread lies in this state until the timeout is completed or until a notification isreceived.
- For example, when a thread calls sleep or a conditional wait, it is moved to timed
  waiting state.

5. Terminated State:

- A thread terminates because of either of the following reasons:
  - Because it exits normally. This happens when the code of thread has entirelyexecuted by the program.

6a **What is autoboxing? Write a Java program that demonstrates how autoboxing and unboxing take place in expression in evaluation.** CO5 L3

- **Autoboxing** is the process by which a primitive type is automatically encapsulated (boxed) into its equivalent type wrapper whenever an object of that type is needed. There is no need to explicitly construct an object.

```
class auto {
public static void
main(String args[]) {
Integer iOb, iOb2; int i;
iOb = 100;
System.out.println("Original value of iOb: " + iOb);      //The
following
                            automatically unboxes iOb, performs
                            the increment, and then reboxes the
                            result back into iOb.
++iOb;
```

```
System.out.println("After
++iOb: " + iOb); iOb2 = iOb +
(iOb / 3);
System.out.println("iOb2 after
expression: " + iOb2); i = iOb + (iOb
/ 3);
System.out.println("i after expression: " + i);
}
   }
```

o

utput:

```
Original
value of
iOb: 100
After
++iOb: 101
iOb2 after
expression:
134 i after
expression:
134
```

6b **What are enumerations? Explain values() and valueOf() methods with an example program.**

- Enumerators contain a list of constant values that apply to a certain type of data, or object.

The values( ) method returns an array that contains a list of the enumeration constants. The
valueOf( ) method returns the enumeration constant whose value corresponds to the string
passed in str. In both cases, enum-type is the type of the enumeration.

```
enum Apple {
Jonathan, GoldenDel, RedDel, Winesap, Cortland
}
class EnumDemo2 {
public static void main(String args[])
{
Apple ap;
System.out.println("Here are all Apple constants:");
// use values()
Apple allapples[] = Apple.values();
for(Apple a : allapples)
System.out.println(a);
System.out.println();
// use valueOf()
ap = Apple.valueOf("Winesap");
System.out.println("ap contains " + ap);
}
}
```

The output from the program is shown here:
Here are all Apple constants:

CO5 L3

	Jonathan GoldenDel RedDel Winesap Cortland ap contains Winesap			