1. What do you understand by thread synchronization and why it is required? Give the syntax of thread synchronization.

Thread synchronization is the process of controlling the access of multiple threads to shared resources so that only one thread can access the resource at a time.

It is required to prevent data inconsistency, race conditions, and unpredictable results in multithreaded programs.

Syntax of synchronization:

Synchronized method:
```
synchronized void display() {
// critical section
}
```

Synchronized block:
```
synchronized(object) {
// critical section
}
```

2. Explain the different ways used for the implementation of thread synchronization.

Thread synchronization in Java can be implemented in the following ways:

1. Synchronized Method:
When a method is declared synchronized, the lock is applied on the object.

2. Synchronized Block:
Only a specific block of code is synchronized instead of the entire method.

3. Static Synchronization:
Uses class-level lock by declaring static synchronized methods.

4. Using Lock Interface:
java.util.concurrent.locks.Lock provides more flexible locking mechanisms.

These methods ensure safe execution of threads.

3. With the help of a suitable example explain the different types of exceptions?

Exceptions are abnormal conditions that occur during program execution.

Types of Exceptions:

1. Checked Exceptions:
Checked at compile time.
Example: IOException

2. Unchecked Exceptions:
Occur at runtime.
Example: ArithmeticException

3. Errors:
Serious problems not handled by programs.
Example: OutOfMemoryError

Example:
int a = 10 / 0; // ArithmeticException

Exception handling improves program reliability.

4. Write a program to implement the concept of thread synchronization.

```
class Table {
synchronized void printTable(int n) {
for(int i=1;i<=5;i++) {
System.out.println(n*i);
}
}
}

class MyThread1 extends Thread {
Table t;
MyThread1(Table t) { this.t = t; }
public void run() { t.printTable(5); }
}

class MyThread2 extends Thread {
Table t;
MyThread2(Table t) { this.t = t; }
public void run() { t.printTable(10); }
}

public class Test {
public static void main(String[] args) {
Table obj = new Table();
new MyThread1(obj).start();
new MyThread2(obj).start();
```

```
}
}
```

The synchronized method ensures only one thread executes at a time.

5. Elucidate the various strategies used for handling exceptions in a multithreaded environment.

Exception handling in multithreaded environment can be done using:

1. try-catch blocks inside run() method
2. Thread.UncaughtExceptionHandler
3. Using finally block for resource cleanup
4. Proper synchronization to avoid runtime exceptions
5. Graceful thread termination

These strategies improve program stability and fault tolerance.

6. Define the following: a. synchronized block b. finally c. Exception d. synchronization

a. synchronized block:
A block that allows only one thread at a time to execute critical code.

b. finally:
A block that always executes whether an exception occurs or not.

c. Exception:
An abnormal condition that disrupts the normal flow of a program.

d. synchronization:
A technique to control access of multiple threads to shared resources.