Q.1: The output for the given program is as follows:

Output:
1
1
2
2
3
3

The output for this program will be the same every time we run this program as, t1.start() and t2.start() only assures starting the execution of the new thread. But, in this case, sleep(500) controls and synchronizes the sequence of the run method and output.

For further detailed compiler logic, refer below steps:

Program starts executing at main method.

1. A new object t1 is created which is instantiated from class TestSleepMethod1 class which inherits 'Thread' class.
2. A new object t2 is created which is instantiated from class TestSleepMethod1 class which inherits 'Thread' class.
3. Compiler goes to t1.start(); and starts thread t1, which automatically invokes run() method.
4. Similarly t2.start(); starts the thread and invokes run() method.
5. Now for t1, value for 'i' is 1 (which is less than 4), t1 sleeps for 500 ms and after sleep, thread t1 wakes up and prints value of i as 1. After printing the value of I, it is incremented by 1.
6. Value for t2 is assigned as 'i=1' (which is less than 4) t2 sleeps for 500 ms, then it wakes up and prints value of i as 1. After printing the value of I, it is incremented by 1.
7. Step 6 and step 7 is repeated alternatively until value of i becomes 4, which is not less than 4 and hence program terminates.

Q.2: The errors in the given code are as follows along with its explanation and correction:

Errors	**Explanation**	**Corrective actions**
class TestJoinMethod1 im plements Thread{	Thread' is a pre-defined class and hence it needs to be 'inherited' instead of implementing.	class TestJoinMethod1 extend s Thread{
public void start{	We cannot override start method	**public void run(){ .....}**
for(int i==1;i<=3;i+)	The valid syntax of for loop is : for(initialization; termination; and iteration). In order to define the initialization, we use the assignment	for(int i=1;i<=3;i+)

	operator. Here, i==1, '==' is logical operator which gives output as boolean value	
for(int i==1;i<=3;i+)	The iteration in for loop is defined by incremental operator which is defined as 'i++'.	for(int i=1;i<=3;i++)
Thread.sleep{500};	Any method name should end with (), therefore, Thread.sleep{500}; is an invalid expression	**Thread.sleep(500){….}**
TestJoinMethod1 t1=new TestJoinMethod1(1,2); TestJoinMethod1 t2=new TestJoinMethod1(2,4,5); TestJoinMethod1 t3=new TestJoinMethod1('sys',1,'a');	Objects t1, t2, and t3 should not accept any arguments as they are instantiated out of default constructor which does not accept any arguments	TestJoinMethod1 t1=new TestJoinMethod1(); TestJoinMethod1 t2=new TestJoinMethod1(); TestJoinMethod1 t3=new TestJoinMethod1();

(Alternative solution)
TestJoinMethod1 t1=new TestJoinMethod1(1,2);
TestJoinMethod1 t2=new TestJoinMethod1(2,4,5);
TestJoinMethod1 t3=new TestJoinMethod1('sys',1,'a');
	If the programmer wants the new objects should be created by passing values, the overloaded constructors have to be created to maintain programmer's logic. (If this program is reused in future where t1, t2 and t3 should accept the arguments the code can be fixed by creating three overloaded constructors in TestJoinMethod1 class	The constructors can be as follows:
		TestJoinMethod1(int a,int b){
		}
		TestJoinMethod1(int a, int b, int c){
		}
		TestJoinMethod1(String a, int b, char c){
		}

t1.run();	t1.run(); should be instead, t1.start() as it is an error to call run() method without calling start() method. Also, start() method automatically invokes run() method.	t1.start();

Correct Code: (Also I have attached '.java' files named : TestJoinMethod1.java)

```
package assignment2;

class TestJoinMethod1 extends Thread{


	public void run(){

		for(int i=1;i<=3;i++){

			try{

				Thread.sleep(500);

			}catch(Exception e){System.out.println(e);}

			System.out.println(i);
		}
	}


	public static void main(String args[]){

		TestJoinMethod1 t1=new TestJoinMethod1();

		TestJoinMethod1 t2=new TestJoinMethod1();

		TestJoinMethod1 t3=new TestJoinMethod1();
		t1.start();


		try{

			t1.join();

		}catch(Exception e){System.out.println(e);}

		t2.start();

		t3.start();
	}

}
```

Q.3: Write a program by using yield(), sleep(), getName(), join() : (Find attached 'ClassStudents.java' file)

A program that picks random student every time from the given set of three student.

```
package assignment2;

class ClassStudents extends Thread {
	public void run(){
		for (int i=0; i<=3 ; i++){
			System.out.println(Thread.currentThread().getName());
			Thread.yield();




			try {
				Thread.sleep(1500);
			} catch (InterruptedException e) {

				e.printStackTrace();
			}

		}
	}


	public static void main(String args[]) throws InterruptedException{

		ClassStudents a = new ClassStudents();
		ClassStudents b= new ClassStudents();
		ClassStudents c = new ClassStudents();


		a.setName("John");
		b.setName("Johny");
		c.setName("Joe");


		a.start();
		b.start();
		c.start();

		b.join();
		c.join();


	}
}
```

1. Yield() : Threads a, b , c are born and are assigned the names as “john”, “Johny” and “joe” After they are created, it invokes run() method. While iterating the for loop, Thread.yeild() is called. Whenever we the program invokes the yield() method, the thread goes into ready que.
2. Join() : Here, b.join() method defines that the other threads will to wait until the execution of thread b.
3. This program will give different output every time it runs as the expected output should give us the random selection of student. As the order of thread execution will be determined by three methods: sleep() , yield() and join().