6. [A] Output of a java program

(i)
```
class Dog{
public static void main(String[] args) {
Dog d = null;
System.out.println(d instanceof Dog);
}
}
```

Ans: false

(ii)
```
class A {
protected void msg() {
System.out.println("Hello java");
}
}

public class Simple extends A {
void msg() {
System.out.println("Hello java");
}

public static void main(String[] args) {
Simple obj = new Simple();
obj.msg();
}
}
```

Ans: Compiler error! But it'll print "Hello java" if protected is removed from the method msg of class A.

(iii)
```
public class Sample {
public static void main(String[] args) {
try {
int data = 100 / 0;
}
catch (ArithmeticException e)
{
System.out.println(e);
}
System.out.println("rest of the code...");
}
}
```

Ans: java.lang.ArithmeticException: / by zero
rest of the code…

(iv)
```
class Animal {
void eat()
{
System.out.println("eating...");
```

```
}
}
class Dog extends Animal {
void bark()
{
System.out.println("barking...");
}
}
class Cat extends Animal {
void meow()
{
System.out.println("meowing...");
}
}
public class TestInheritance3 {
public static void main(String[] args) {
Cat c = new Cat();
c.meow();
c.eat();
c.bark();
}
}
```

Ans: Compiler error (bark method not available).
Without c.bark() it will print meowing… and eating…

6[B] How to access package from another package?
Ans: In java to access package from another one, we use the import command. For example,

```
import packagename.classname;
```

6[C] What is the purpose of join method?
Ans: join method is used when one thread waits for an another thread to finish it's job. For example, if we want to finish the process of ThreadB and then continue, we can use,

```
ThreadB.join();
```

6[D] How to perform two tasks by two threads?
Ans: To perform two task, we can two separate threads and assign each of them two different task and start them. In this way those two will start working simultaneously.

6[E] What is the Thread Scheduler and what is the difference between preemptive scheduling and time slicing?

A component of Java that decides which thread to run or execute and which thread to wait is called a **thread scheduler in Java**. In Java, a thread is only chosen by a thread scheduler if it is in the runnable state. However, if there is more than one thread in the runnable state, it is up to the thread scheduler to pick one of the threads and ignore the other ones.

- **Preemptive scheduling** allows the scheduler to take a running thread away from the CPU and give it to another thread, even if the first thread has not finished its execution. This is done

based on the priority of the threads. A thread with a higher priority will preempt a thread with a lower priority.

- **Time slicing** is a type of preemptive scheduling where each thread is given a certain amount of time to run on the CPU. After the time slice expires, the thread is preempted and another thread is given a chance to run. This process continues until all of the threads have had a chance to run.

The main difference between preemptive scheduling and time slicing is that in preemptive scheduling, the scheduler can preempt a running thread at any time, while in time slicing, the thread is only preempted after its time slice expires.