2(a)

implements Runnable is better. However, the caveat is important

- Java doesn't support multiple inheritances, which means you can only extend one class in Java so once you extended Thread class you lost your chance and cannot extend or inherit another class in Java.
- In Object-oriented programming, extending a class generally means, adding new functionality, and modifying or improving behaviors. If we are not making any modification on Thread then use Runnable interface instead.
- Runnable interface represents a Task which can be executed by either plain Thread or Executors or any other means. so logical separation of Task as Runnable than Thread is a good design decision.
- Separating task as Runnable means we can reuse the task and also has the liberty to execute it from different means. since you can not restart a Thread once it completes. again Runnable vs Thread for task, Runnable is winner.
- Java designer recognizes this and that's why Executors accept Runnable as Task and they have worker thread which executes those task.
- Inheriting all Thread methods are additional overhead just for representing a Task which can be done easily with Runnable.

With `implements Runnable`:

```
public class MyRunnable implements Runnable {
    public void run() {
        //Code
    }
}
//Started with a "new Thread(new MyRunnable()).start()" call
```

Or, with `extends Thread`:

```
public class MyThread extends Thread {
    public MyThread() {
        super("MyThread");
    }
    public void run() {
        //Code
    }
}
//Started with a "new MyThread().start()" call
```