```
    stream.mark();
  }

  @Override
  public void reset() throws IOException {
    throwIfClosed();
    stream.reset();
  }

  @Override
  public void close() throws IOException {
    isOpen = false;
    stream.close();
  }

  private void throwIfClosed() {
    Preconditions.checkState(isOpen, "Serializer is closed!");
  }

  public static class ProtobufDeserializerBuilder implements Builder {

    @Override
    public EventDeserializer build(Context context,
      ResettableInputStream resettableInputStream) {
      // The serializer does not need any configuration,
      // so ignore the Context instance. If some configuration has
      // to be passed to the serializer, this Context instance can be used.
      return new ProtobufDeserializer(resettableInputStream);
    }
  }
}
```

56

ProtobufDeserializer 类从文件中读取 Protobuf-serialized 事件，并在 readEvent 方法中将它们转换为 Flume 事件，返回为 null 时表示没有可用的事件读取。如果 readEvents 方法没有事件可以读取，将返回一个空列表，就如 EventDeserializer 接口强制要求的那样。

由于文件是不可变的，一旦我们达到一个阶段，即文件中没有更多的可用的事件，这意味着所有的事件已经从文件读取完，此时 Source 通过调用 close 方法关闭反序列化器。如果序列化器保持任何内部状态或有一些清理工作要做，这种方法有望做到这一点。在这种情况下，我们只是简单地关闭流。mark 和 reset 方法只是检查反序列化器是否打开和转发调用方法给流。序列化器特定的实现不需要任何配置，但反序列化器可以通过