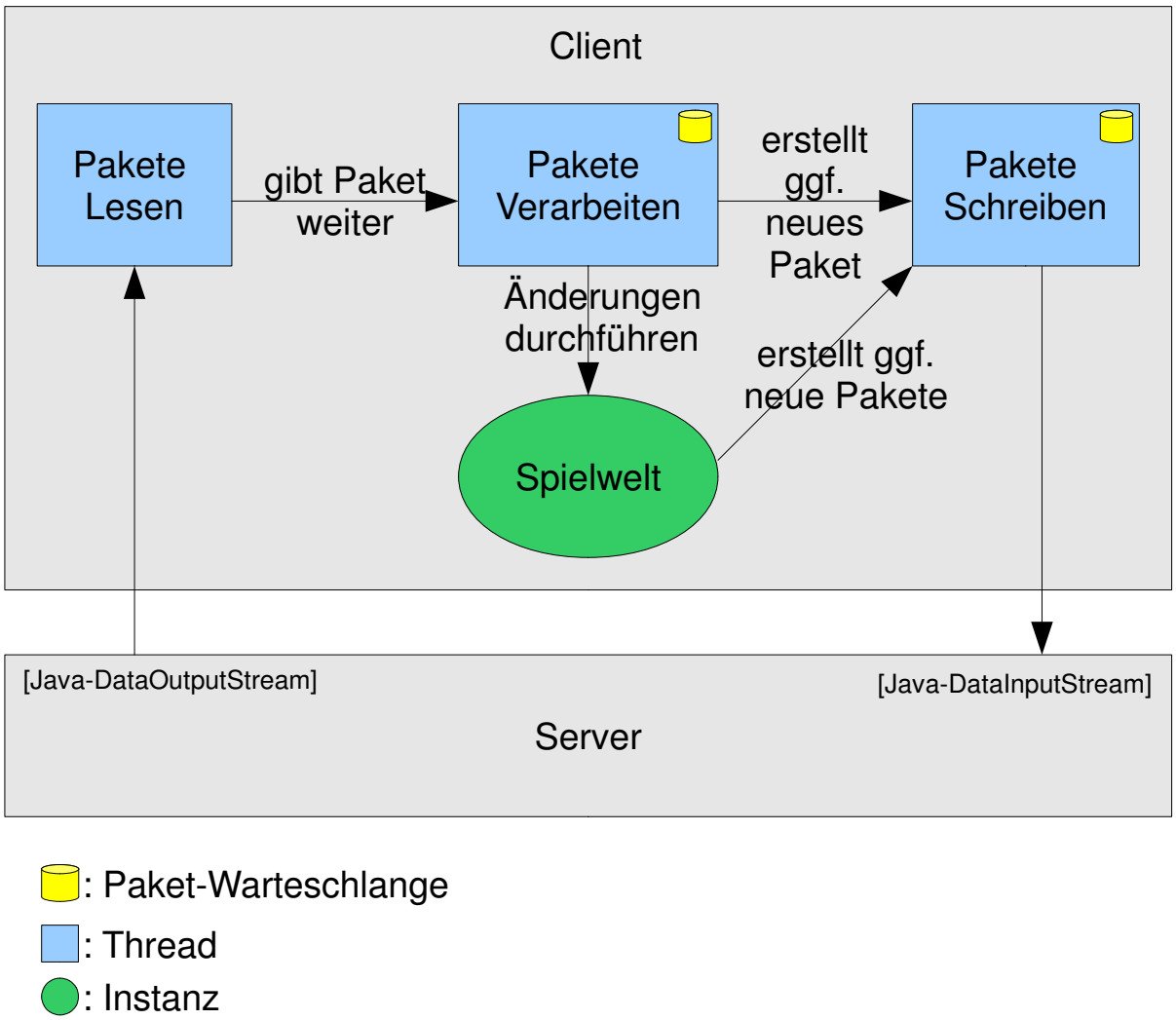
Client / Server – Kommunikation

Client und Server sind über TCP verbunden. Als Kommunikationsschnittstelle nutzt der Server das DataOutputStream und DataInputStream von Java. Der Client muss dem nach über eine zu Java kompatible Kommunikationsschnittstelle verfügen. Dabei können Probleme auftreten zum Beispiel bei unterschiedlichen Längen von Datentypen oder der Endian-Konvention.

Die Kommunikation läuft gruppiert in Paketen ab. Jedes Paket beginnt mit 1 Byte das es spezifiziert. Daran ist also zu erkennen um welches Paket es sich handelt. Daraufhin folgt je nach Paket eine bestimmt Anzahl an Bytes. Für jedes Paket ist definiert welche Daten gelesen/geschrieben werden müssen und um was für Daten es sich handelt.



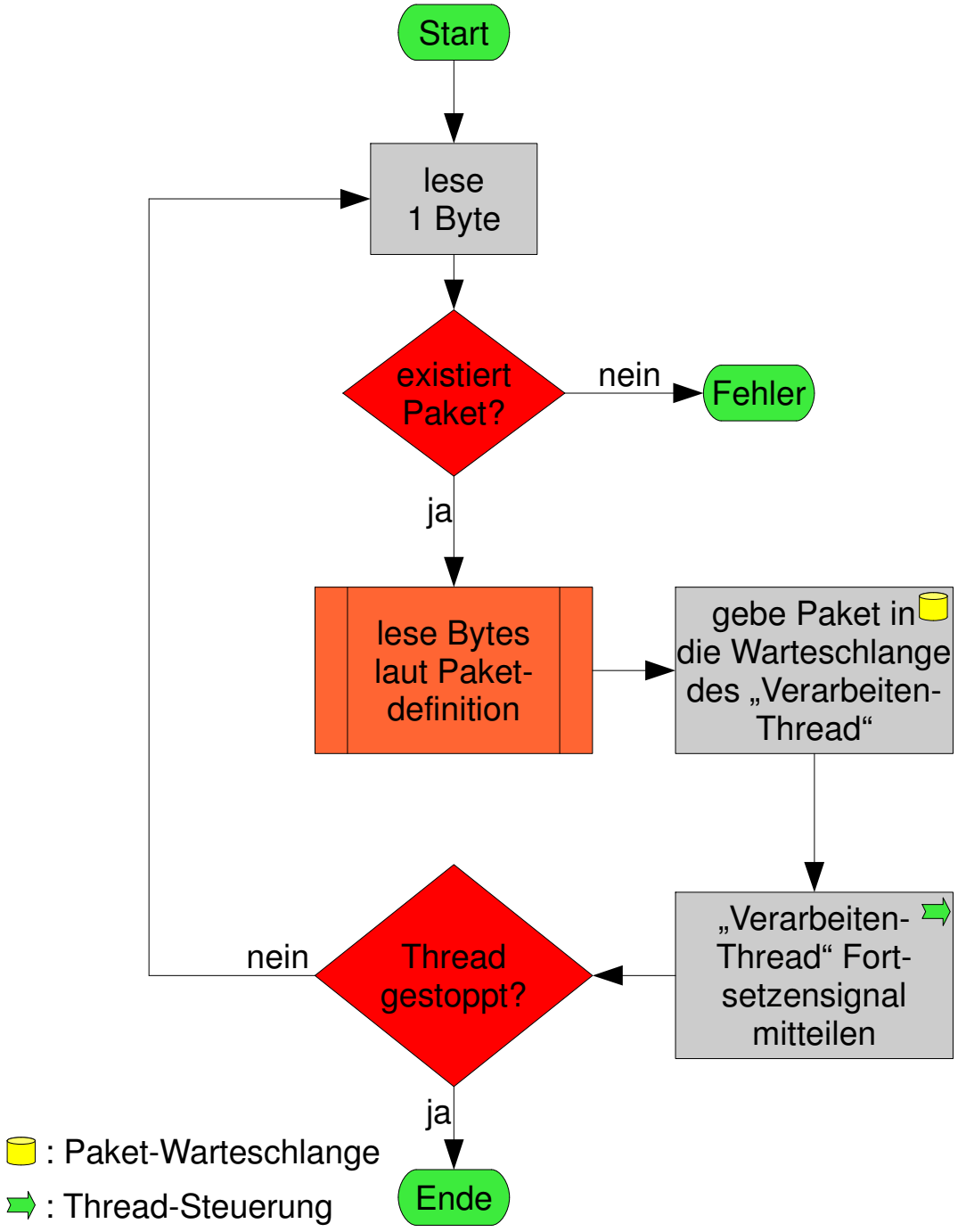
: Paket-Warteschlange

: Thread

: Instanz

Pakete Lesen – Thread

Der "Pakete Lesen Thread" liest sequenziell die vom Server reinkommenden Paket ein und gibt sie an den "Pakete Verarbeiten Thread" ab. Die Paket werden nach dem Lesen nicht im selben Thread verarbeitet, weil dadurch das Lesen weiterer Pakete blockiert wird und der Server dann die Verbindung mit Buffer-Overflow trennt.



Pakete Verarbeiten – Thread

Dieser Thread besitzt eine Prioritätswarteschlange (priority_queue) von Paketen. Sie wird vom "Pakete Lesen Thread" gefüllt. Die Warteschlange wird in Reihenfolge der Priorität abgearbeitet. Minecraft an sich sieht keine priorisierte Behandlung von Paketen vor. Dies ist aber notwendig um auf langsameren Geräten eine kontinuierliche Verbindung Aufrecht zu erhalten. Zum Beispiel muß auf das "KeepAlive"-Paket innerhalb einer gewissen Zeit geantwortet werden, sonst trennt der Server die Verbindung. Wenn die Warteschlange des Threads leer ist, wird gewartet bis er vom "Paket Lesen Thread" ein Signal zum fortsetzen bekommt.

Pakete Schreiben – Thread

Wie auch der "Pakete Verarbeiten Thread" besitzt dieser Thread, aus dem selben Grund, auch eine Prioritätswarteschlange.