

- jre** Die vollständige Java-Laufzeitumgebung (Java Runtime Environment), also die „Java Virtual Machine“ (JVM). Diese ist für die eigentliche Ausführung von Java-Programmen erforderlich.
- macros** Unterordner für ImageJ-Makros (sind hier nicht behandelt).
- plugins** In diesem Ordner werden die eigenen ImageJ-Plugins gespeichert. Er enthält bereits einige andere Unterordner mit Beispiel-Plugins (Abb. C.1 (b)). Eigene Plugins dürfen nicht tiefer als eine Verzeichnisebene unter diesem Ordner liegen, da sie ansonsten von ImageJ nicht akzeptiert werden.
- ij.jar** Eine „Java Archive“-Datei, in der die gesamte Basisfunktionalität von ImageJ enthalten ist. Bei einem Update auf eine neuere Version von ImageJ muss i. Allg. nur diese eine Datei ersetzt werden. JAR-Dateien enthalten Sammlungen von binären Java-Files (.class) und können wie ZIP-Files geöffnet werden.
- IJ_Prefs.txt** Eine Textdatei, in der diverse Optionen für ImageJ eingestellt werden können.
- ImageJ.cfg** ... Enthält die Startparameter für die Java-Laufzeitumgebung, im Normalfall die Zeile
`"jre\bin\javaw.exe -Xmx300m -cp ij.jar ij.ImageJ"`.
 Die Option `-Xmx300m` bestimmt dabei, dass anfangs 300 MB Speicherplatz für Java angefordert werden. Diese Größe ist für manche Anwendungen zu klein und kann an dieser Stelle modifiziert werden.
- ImageJ.exe** ... Ein kleines Launch-Programm, das über Windows wie andere Programme gestartet wird und das anschließend selbst Java und ImageJ startet.

Um eigene Plugin-Programme zu erstellen, ist außerdem ein Texteditor zum Editieren der Java-Files und ein Java-Compiler erforderlich. Das in ImageJ verwendete Java Runtime Environment enthält beides, also auch bereits einen Compiler, sodass grundsätzlich keine weitere Software notwendig ist. Die so verfügbare Programmierumgebung ist aber selbst für kleinere Experimente unzureichend und es empfiehlt sich die Verwendung einer zusätzlichen Java-Programmierumgebung, wie beispielsweise *Eclipse*¹ oder *Borland JBuilder*². Damit ist insbesondere bei umfangreicheren Plugin-Projekten eine saubere Projektverwaltung und Programmanalyse möglich, mit der viele Programmierfehler bereits im Vorfeld zu vermeiden sind, die andernfalls erst während der Programmausführung auftreten.

¹ www.eclipse.org.

² www.borland.com/jbuilder.