BlueJ - eine didaktische Java-Entwicklungsumgebung

Autor
Jürgen Obermeyer
Universität Oldenburg - Fachbereich Informatik
D-26129 Oldenburg - Germany
Email: obermeyer@informatik.uni-oldenburg.de

Zusammenfassung: Eine integrierte Entwicklungsumgebung (Integrated Development Environment - IDE) dürfte heute fester Bestandteil der Werkzeugpalette eines jeden professionellen Softwareentwicklers sein. Sie fasst verschiedene Tools wie Editor, Compiler oder Debugger unter einer einheitlichen Oberfläche zusammen und erleichtert damit die Programmierarbeit z.T. erheblich. Ihr komplexer Aufbau und vor allem ihre Funktionsvielfalt lassen es jedoch fraglich erscheinen, ob IDEs auch dazu geeignet sind, Schülerinnen und Schüler beim Erlernen des Programmierens zu unterstützen. Genau diesen Anspruch aber hat BlueJ, eine integrierte Java-Entwicklungsumgebung.

1 Einleitung: Integrierte Entwicklungsumgebungen

Integrierte Entwicklungsumgebungen gehören heute zum Handwerkszeug eines jeden Entwicklers. Sie führen zahlreiche Programme, die während der Implementierungsphase einer Software benötigt werden, unter einer einheitlichen Oberfläche zusammen. Ihre Funktionen sind sehr vielfältig; ihre Mächtigkeit und damit auch ihre Komplexität nimmt von Version zu Version zu. In einem Artikel über das Open-Source-Projekt Eclipse heißt es:

> "Software-Entwickler erwarten von einer modernen Entwicklungsumgebung mehr als nur einen einfachen Editor, Compiler und Debugger. Sie sollte Komfortfunktionen wie automatische Code-Ergänzung und Klassen-Browser bieten, aktuelle Entwicklungsmethoden unterstützen und sich leicht an die individuellen Anforderungen eines Entwickler-Teams anpassen lassen." [A03, S. 196].

Die Firma "Brockhaus AG - Software & Consulting" fasst den Begriff "Integrierte Entwicklungsumgebung" noch weiter:

> "Bei der Brockhaus AG gehören zu diesem wichtigen Instrument ein Versionskontrollsystem, ein Testsystem und ein System zur Komponentenzusammenstellung. [...] Darüber hinaus nutzen wir die Fähigkeiten moderner Entwicklungsumgebungen zur kontinuierlichen Qualitätssicherung, z. B. zum Refactoring und zur automatischen Prüfung unserer Quellcodekonventionen." [K03].

Für den Profi ist der Einsatz derartiger Tools natürlich verlockend:

- viele häufig genutzte Funktionen sind nur einen Mausklick weit entfernt;
- die Programme sind sehr individuell konfigurierbar und passen sich somit dem persönlichen Geschmack des Benutzers weitgehend an;
- in verschiedenen Fenstern wird gleichzeitig eine Fülle von Informationen geboten: im Editorfenster der Quellcode, im Ausgabefenster Compilermeldungen oder Suchergebnisse, in der Projektverwaltung eine Übersicht der zum Projekt gehörigen Klassen, im Klassenfenster die Liste aller Variablen und Methoden usw.

Fraglich ist, ob diese vom Profi geschätzte Vielfalt auch dazu angetan ist, Schülerinnen und Schüler in ihrem Prozess des Erlernens einer Programmiersprache zu unterstützen. Selbst bei einer kleineren Entwicklungsumgebung wie dem *JCreator LE* ist das Hauptfenster in vier verschiedene Bereiche unterteilt: die Projektverwaltung (in der zudem zwischen "Workspaces" und "Projects" unterschieden wird), den Editor, die Paketansicht und das Ausgabefenster für Compilermeldungen. Ferner werden neben der üblichen Menüleiste zwei Werkzeugleisten angezeigt, die insgesamt 44 (!) verschiedene Buttons enthalten.

Dieser Informationsüberfluss lenkt stark vom eigentlichen Ziel ab und erschwert insbesondere den Einstieg in die Programmierung. Da Entwicklungsumgebungen zudem nicht unter didak-

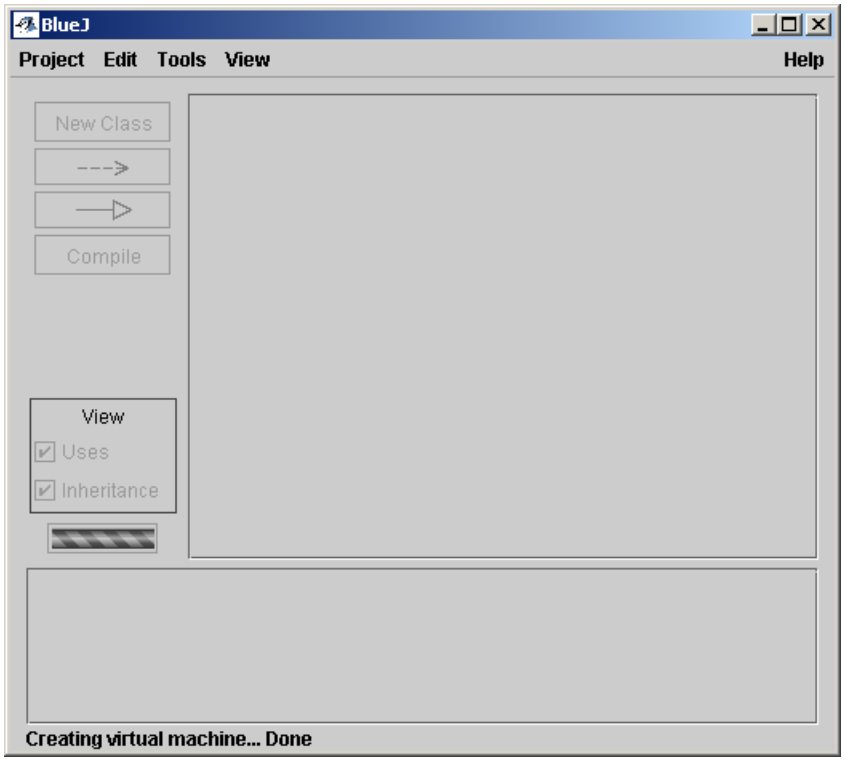
tischen Gesichtspunkten entworfen werden, einen oft beachtlichen Ressourcenhunger an den Tag legen und überdies z.T. mit hohen Lizenzgebühren belegt sind, ist von ihrem Einsatz in der Schule eher abzuraten.

2 BlueJ, eine didaktische Java-Umgebung

Die Java-Entwicklungsumgebung BlueJ ist ein (Teil-) Produkt eines Forschungsprojekts der Universitäten Melbourne und Süd-Dänemark. Ziel dieses Projekts ist es, Werkzeuge und Methoden zu entwickeln, mit deren Hilfe man Anfängern das Erlernen der Grundkonzepte objektorientierter Softwareentwicklung erleichtern kann. In Bezug auf die Gestaltung von BlueJ sind vier Aspekte von zentraler Bedeutung:

> "I. **BlueJ is object-oriented.**
> In BlueJ students interact with classes and objects. They can manipulate class structure graphically and textually. Objects can be created and methods of any object can be called interactively.
> II. **BlueJ has been designed for teaching.**
> BlueJ offers a unique mix of sophisticated support for visualisation and interaction and a simple and intuitive interface.
> III. **BlueJ visualises class structure and objects.**
> In BlueJ, class structure can be seen and manipulated on screen, and objects can be interactively created and operations called.
> IV. **BlueJ is free.**
> The environment is available free of charge." [Kölling 2002]

Die Programmoberfläche, mit der sich BlueJ nach dem Start präsentiert, wirkt im Vergleich zu den o.g. professionellen Entwicklungsumgebungen geradezu minimalistisch (siehe nebenstehende Abbildung). Eine Menüleiste mit fünf Einträgen sowie vier deaktivierte Schaltflächen - viel mehr wird dem Anwender zunächst einmal nicht zugemutet. Die Autoren sind hier ganz klar nach dem Motto "Weniger ist mehr!" verfahren, ohne dabei jedoch den Funktionsumfang, den man von einer Entwicklungsumgebung erwartet, wesentlich zu beschneiden.

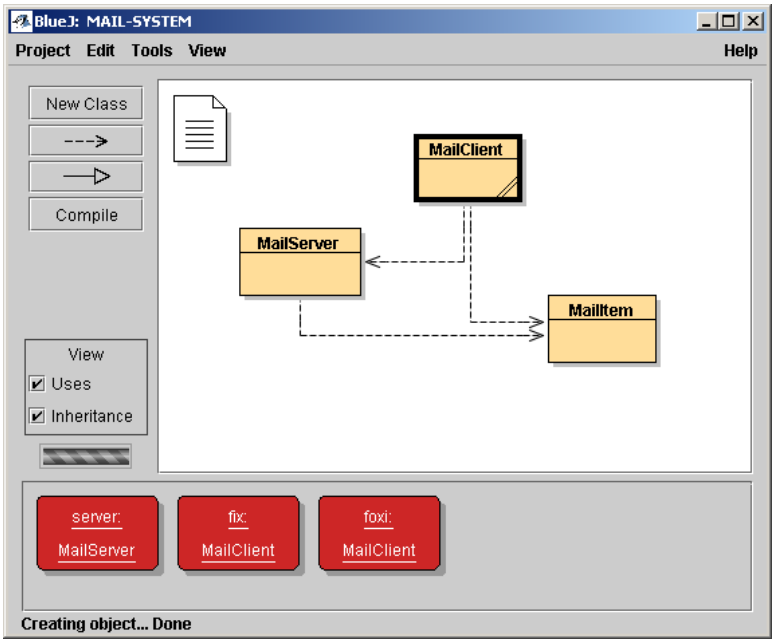


BlueJ bietet u.a. ...

- ... eine grafische Anzeige der Klassenstrukturen in Form reduzierter UML-Klassendiagramme;
- ... grafisches und textuelles Editieren;
- ... einen eingebauten Editor, Compiler, Debugger;
- ... die interaktive Erzeugung von Objekten;
- ... interaktive Methodenaufrufe;
- ... interaktives Testen;
- ... eine Projektverwaltung.

Neben der didaktisch geschickt reduzierten Oberfläche sprechen vor allem zwei weitere Vorteile für den Einsatz von BlueJ in einer schulischen Lernumgebung:
1. Den beiden Hauptproblemen, denen sich Java-Anfänger ausgesetzt sehen, wird wirksam begegnet: mit BlueJ besteht weder die Notwendigkeit, vor der Ausführung eines Programms eine `main`-Methode schreiben zu müssen (allein deren Signatur greift auf Konzepte zurück, die Anfängern nicht zu erklären sind), noch müssen sich die Schülerinnen und Schüler um die Ein- und Ausgabe von Text kümmern, die unter Java alles andere als trivial ist. Objekte können interaktiv erzeugt (per Rechtsklick auf das Klassendiagramm) und Methoden interaktiv aufgerufen werden (per Rechtsklick auf ein Objekt), wobei auch Parameter übergeben werden können. Eventuelle Ausgaben werden in Textboxen dargestellt.

2. BlueJ arbeitet sehr stark mit **Visualisierungen**:
"Many Java programming concepts are abstract, such as classes and objects, methods and fields. […] Teaching diagrams (with text) help illustrate object-oriented concepts, but seeing the process at work makes it that much easier to learn. BlueJ does just that by diagramming classes and objects in UML-like format. [...] In addition, the diagram shows the relationships between classes, so students have a clear understanding of fundamental OOP concepts before they've even seen any code. This allows teachers to take the Object first concept into reality without having to confuse students with syntax details. The simple subset of UML diagram notation allows for migration to full UML in later courses as the student progresses. " [Nourie 2002]



Weitere Möglichkeiten bestehen im Einsatz des *Inspectors* bzw. des *Debuggers*. Mit dem *Inspector* lässt sich der aktuelle Zustand eines Objekts genau untersuchen; der Debugger hilft durch das Setzen von Breakpoints und durch die schrittweise Abarbeitung von Methoden, bestimmte Konzepte (z.B. die verschiedenen Alternativen bei Auswahlabfragen) besser zu verstehen bzw. Programmierfehler zu finden.

3 Objects first with Java

Im Herbst 2002 haben die BlueJ-Autoren ein Lehrbuch veröffentlicht, das vollständig auf BlueJ aufbaut ist und in insgesamt 13 Kapiteln mehr als nur einen Überblick über die objektorientierte Programmierung in Java vermittelt. Zu jedem Kapitel werden ansprechende, motivierende BlueJ-Projekte mitgeliefert, deren besonderer Reiz darin besteht, dass die Schülerinnen und Schüler von der ersten Stunde an mit ihnen arbeiten können, ohne bezüglich der Aufgabenstellung ins Triviale abzugleiten. So ist *Objects First With Java* einer der ganz wenigen Java-Lehrgänge, der nicht mit dem ebenso berühmten wie berüchtigten "Hello world!"-Problem beginnt. Stattdessen wird sehr schnell mit mehreren Objekten (auch Objekten verschiedener Klassen gearbeitet), damit die Schülerinnen und Schüler verstehen, dass sinnvolle Applikationen nur dann zu Stande kommen, wenn viele Objekte miteinander interagieren.
Allerdings soll nicht verschwiegen werden, dass in der Verwendung der vorgefertigten Projekte auch eine Gefahr liegt: zwar müssen diese von den Schülerinnen und Schülern immer wieder analysiert und verändert werden, jedoch äußern sie sehr schnell den Wunsch, auch einmal vollständig eigene Klassen zu schreiben. Dafür bieten sich dann Übungsphasen an, in denen die

bisher behandelten Projekte in abgewandelter Form "nachgebaut" oder aber vollständig neue Projekte erstellt werden.

Literatur

[Aust 2003] AUST, Stefan Matthias; RUSITSKA, Michael: *Offener Werkzeugkasten. Java-Software entwickeln mit Eclipse*. In: c't magazin für computer technik, Ausgabe 3/2003, S. 196-201.

[Barnes 2002] BARNES, David; KÖLLING, Michael: *Objects First with Java. A Practical Introduction using BlueJ*. Harlow: Pearson Education Limited 2002.

[Kölling 2002] KÖLLING, Michael: *Why BlueJ? An introduction to the problems Bluej addresses.* http://www.bluej.org/why/why.html [geprüft: 10.02.2003].

[Kloss 2003] KLOSS, Michael: *Integrierte Entwicklungsumgebung.* http://www.brockhaus-ag.de/bag_live/bag/DE/SoftwareFactory/Entwicklungsumgebung/index.jsp [geprüft: 10.02.2003].

[Nourie 2002] NOURIE, Dana: *Teaching Java Technology With BlueJ.* http://java.sun.com/features/2002/07/bluej.html [geprüft: 10.02.2003].