Objekt

Objekte: Zustand (Attribute), Identität, Lebensdauer, Verhalten, Erzeugung (Konstruktor)

Ausdruck, Objektreferenz, Variable, Alias, Zuweisung

Nachricht, Methode, Empfängerobjekt, impliziter Parameter

Schnittstelle

Klassifikation nach Schnittstelle

Klasse, abstrakte Klasse, Interface → Vorgriff auf Typen

Vererbung vs. Delegation, Mehrfachvererbung

Spezialisierung, Abstraktion

Überschreiben

Binden: statisch vs. dynamisch

Statische Bindung von Attributen in Java

Verdecken bzw. Verstecken von Attributen

this, super

Typen

Subtypbeziehung, ist-ein-Beziehung (Abgrenzung zur ist-Instanz-Beziehung)

Liskovsches Substitutionsprinzip, Ko- und Kontravarianz

Typsicherheit

Lücken bei statischer Typsicherheit in Java: Rolle von null, ArrayStoreException…

Statischer Typ vs. dynamischer Typ

dynamische Methodenwahl

Cast

Polymorphie, Subtypolymorphie

parametrische Polymorphie

beschränkte parametrische Polymorphie

beschränkte parametrische Polymorphie mit Wildcards

Überladung (ad-hoc-Polymorphie), Auflösung von Überladung, Most-Specific-Algorithmus

Auflösung eines Methodenaufrufs (statisch, dynamisch), Bytecode

Klassenmethoden, Klassenattribute, Rolle von static, Verdecken von Klassenmethoden

Abgrenzung Verdecken – Überschreiben

Abgrenzung Subtyping – Subclassing – Vererbung

Warnung vor „static“,

nonstatic vs. static: „gehört zu“ vs. „ist zugeordnet zu“, was heißt das für Attribute, Methoden, innere Klassen? → FAQ!

Exception
Rolle des „Parameters“, „selektives“ Fangen
„catch or declare“
checked exception vs. unchecked exception (RuntimeException)
try / catch / finally
throw-Anweisung vs. throws-Deklaration
Exceptions sind typischerweise KEINE inneren Klassen!
Stacktrace erläutern

Kapselung
Sichtbarkeit, private, package, protected, public
Pakete
innere Klassen, statische innere Klassen (Warnung vor sinnfreiem Gebrauch!)

Aufzählungstypen (Enums), Enums sind typischerweise KEINE inneren Klassen!

lokale Klassen

allgemein: Code zur späteren Ausführung übergeben (Block, First-Class-Method, Objekt, Closure), Instanzen anonymer lokaler Klassen

Ströme, allgemein und dann noch einmal Objektgeflechte am Chareingabestrom-Beispiel

AWT
AWT als Programmgerüst (Framework), Definition Programmgerüst, Abgrenzung
Layoutmanager
Beobachter, Ereignisse, Parallelität beim AWT (Event Dispatch Thread, Ereigniswarteschlange)
Listener: Weckdienstbeispiel
Verknüpfung von Elementen einer graphischen Benutzeroberfläche
MVC, Wie „verheirate“ ich Objekte? Standardfall (Übergabe im Konstruktor) am Beispiel vorexerzieren

Parallelität

Thread vs. Runnable

start() vs. run()

Abgrenzung BS-Thread vs. Exemplar der Klasse Thread

Scheduling

Inkonsistenzen durch Parallelität

Synchronisation, Monitor (Eisenbahnbeispiel), Threadzustände-Diagramm

Synchronizes Methode vs. synchr. Block mit expliziter Angabe des Synchronisationsobjekts („Kontrollstation“)

Synchronisation sperrt keine Objekte, sondern Monitore

Mehrfachsperrung durch denselben Thread

Deadlock

Erzeuger/Verbraucher mit wait() / notify() / notifyAll()

Verteilte Systeme

Entfernter Methodenaufruf

Stub-Skeleton

Socket

Client-Server

Schlüsselwörter/Modifier: Wo sind static, final, synchronized, abstract, private… erlaubt und was bedeutet das dort jeweils?

Immer wieder ärgerlich: Punktabzug wegen Fehlern bei der Formulierung

- Er (Compiler) – Sie (Laufzeitumgebung)
- Passendste vs. speziellste passende Methode
- Anweisungen stehen immer nur in Blöcken (Methoden, Konstruktoren, Initializer), Ausnahme: Attributdeklaration mit direkt folgender Initialisierung
- Klassen und Objekte werden nicht „aufgerufen“, aufgerufen werden Methoden und Konstruktoren. Klassen werden ggf. instantiiert, Objekten werden Nachrichten geschickt. Methoden werden *auf* Ausdrücken aufgerufen.