PM2 Zusammenfassung

* Typkonvertierung
  + Narrowing, Downcasting
  + Widening, Upcasting
* Assoziativität
  + Rechtsassoziativ: Auswertung von rechts nach links (++, --, !)
  + Linksassoziativ: Auswertung von links nach rechts (+, -, \*, <, &&)
* Varargs
  + Variable length argument list
  + void sum(int… args) { }
* Kovarianz
  + Typkompatibilität zwischen Arrays, die sich aus der Kompatibilität der Komponententypen ableitet
  + Ungeschützt
    - Object[] oa = new String[3]; oa[0] = 4
    - Kein Compilerfehler, aber Laufzeitfehler
* Arrays befüllen
  + Arrayliterale
  + Iteration mit *for* und index
  + Für *char[]* mit *toArray, getChars* von *String* und *StringBuffer*
  + *toArray*-Methode der *Collection* Klassen
* Arrays kopieren
  + Explizites Übertragen in Schleifen
  + *Clone* der Klases *Object*
  + *copyOf* der Klasse *Arrays*
  + *System.arraycopy*
* Arrays vergleichen
  + *equals,* Referenztypen
  + *==,* primitive Objekte
  + *deepEquals,* mehrdimensionale Arrays
* Kopien
  + Flach
  + Tief
* Vergleich
  + *==*, Identität
  + *Equals*, Inhaltsgleichheit (immer nutzen!)
  + Compiler kann aus inhaltsgleichen String einen machen aus Platzgründen 🡪 Stringpool
* Sichtbarkeit
  + *public, protected, package private* (Default)*, private*
  + Klassen ohne *package-*Deklaration 🡪 Default Package
    - Diese Klassen können nicht importiert werden
  + Packages bilden keine Hierarchie
  + Import
    - Statisch: Klassenvariablen und -methoden können ohne den Klassennahmen verwendet werden
* Streams
  + Datenverarbeitungs-Pipeline
  + Lambda-Ausdrücke oder Methodenreferenzen als Argumente
  + Kein indizierter Zugriff, aber Umwandlung in Listen und Arrays
  + Eigenschaften, die Listen fehlen
    - Lazy-Evaluation
    - Parallele Verarbeitung
    - Potentiell unendlich
* Methoden
  + Überschreiben: Methode einer abgeleiteten Klasse ersetzt die Definition einer Methode gleicher Signatur der Superklasse
    - Methode der Superklasse nicht mehr erreichbar, außer mit super.\*
  + Überladen: Methode einer Klasse gleichen Namens, aber ungleicher Signatur der Klasse oder einer der Superklassen definiert eine neue Methode
    - Methoden gleichen Namens, aber ungleicher Signatur in der Klasse oder deren Superklassen sind weiterhin erreichbar
* Generische Typen
  + Invarianz: C<T>
    - Typargumente vom selben Typ
    - Lesen, Schreiben zulässig
  + Kovarianz: C<? extends B>
    - Wildcard-Typen mit Upper-Typebound
    - Alle generischen Typen, deren Typargument zu B kompatibel sind zum Wildcard-Typ
    - Lesen zulässig
    - Object[] oa = new String[3]; oa[0] = 4;
  + Bivarianz: C<?>
    - Wildcard-Typen ohne Einschränkungen
    - Alle generischen Typen der gleichen generischen Klasse wie der Wildcard-Typ sind kompatibel zum Wildcard-Typen
  + Kontravarianz: C<? super B>
    - Wildcard-Typen mit Lower-Typebounds
    - Alle generischen Typen, deren Typargument Supertyp von B ist, sind kompatibel zum Wildcard-Typen
    - Schreiben zulässig
* Generische Array-Objekte
  + T[] elements; elements = (T[])new Object[5]; im Konstruktor 🡪 schlechter Stil
    - Lösung: Erzeugen des Arrays und das Festlegen des Komponententyps auf die Laufzeit verschieben
  + Zugriffsmethoden müssen beim Setzen und Lesen den Komponententyp sicherstellen