Kontrollfragen A

1. Wie steht es hier mit der Gleichheit?
   Entity Types = gleiche Identität (einzigartig, es gibt nur ein Objekt)
   Value Types = gleiche Werte (mehrere Objekte die gleich sind)
   Zugehörigkeit = gleicher Klassentyp (mehrere Objekte von gleichen Klassentyp, jedoch unterschiedliche Werte)
   - Bücher
     Value Types und Entity Types sind möglich
   - Prozesse
     Entity Types (ein Prozess ist eindeutig, z.B. wegen der Prozess-ID)
   - Stacks
     Entity Types (eindeutig, es gibt nur ein einziger Stack und nicht mehrere)
   - GUI-Komponenten
     Entity Types (jede Komponenten muss einzeln angesprochen werden können, sonst wäre z.B. nicht klar auf welchen Button gedrückt wurde)
   - Verträge
     Entity Types (Jeder Vertrag muss eindeutig sein, z.B. durch die eindeutige Vertragsnummer)

2. Im Falle von Identität haben wir es mit der stärksten Form von Gleichheit zu tun. Welches ist die nächstschwächere Form?
   Value Types
   (Entity Types -> Value Types -> gleicher Klassentyp)

Kontrollfragen B

1. Welche Objekte besitzen eine Methode `equals()`?
   Alle (alle Objekte sind vom Typ Object abgeleitet, welche mitunter die Methode equals() implementiert hat!)

2. Wie ist `equals()` defaultmässig implementiert?
   Entity Types, Identitätstest
   ```
   public boolean equals(Object obj)
   {
     return (this == obj);
   }
   ```

3. Bestimmte Java-Klassen sind darauf angewiesen, dass `equals()` richtig implementiert ist. Nennen Sie Beispiele.
   Grundsätzlich alle Collections-Klassen / Datenstrukturen, z.B. Array, Set, LinkedList, etc.

4. Der equals()-Contract schreibt symmetrisches Verhalten vor. Was heisst dies?
   x.equals(y) und y.equals(x) müssen das gleiche Resultat ergeben (x=y und y=x)

5. In welchen 4 Schritten implementiert man in der Regel `equals()`?
   1. *Test auf Identität (Alias-Prüfung)*
   2. *Test auf null*
   3. *Test auf Typen-Vergleichbarkeit*
   4. *Vergleich aller relevanten Felder (== resp. mittels equals())*

6. Wieso braucht's auf der vorherigen Folie im 4. Schritt ein Cast?
   Das Objekt wird vom Typ Object übergeben, Object kennt die spezifischen Methoden nicht. => Cast auf Balloon

7. Was könnten zusätzliche Attribute für `Balloon` sein?
   Size

Kontrollfragen C

1. Weshalb muss man `hashCode()` häufig überschreiben?
   Das Verhalten von hashCode() muss mit jenem von equals() korrespondieren.
   Im Idealfall müssen für ungleiche Objekte unterschiedliche Hash-Codes erstellt werden. Praktisch kann aber für mehrere Objekte derselbe Hash-Code resultieren.

2. Zwei Objekte sind gemäss `equals()` ungleich. Muss `hashCode()` für diese zwei Objekte auch ungleiche Werte liefern?
   Nein, nicht zwingend. Im Idealfall jedoch schon!

3. Wieso lassen sich Datenstrukturen mit Objekten, welche mit der Default-Implementierung von `hashCode()` arbeiten, nicht persistent speichern?
   In der Default Implementation von hashCode() wird die aktuelle Adresse des Objekts auf dem Heap zurückgegeben. Dieser ist nicht statisch und somit nicht „persistent". Ein „serialization" ist somit nicht möglich.

4. Auf der vorletzten Folie fügt man 4 Ballone in `Set` ein. Offenbar sind aber nur 2 darin! Wieso?
   b1 wird zweimal eingefügt, sowie b2 ist gleich b1 – dies gibt jeweils eine Kollision und sie werden deswegen nicht gespeichert. (resp. je nach Implementation als „Bucket" unter dem gleichen Index gespeichert)