Aufgabe 1.1a)

* Frage 1: Wie können Sie unter Berücksichtigung der Prinzipien des objektorientierten Entwurfs dafür sorgen, dass der Code, der den beiden Implementierungen gemeinsam ist, nicht dupliziert wird?

|  |
| --- |
| Man könnte das Interface NumberTransformer.java zu einer abstrakten Klasse abändern und dort eine Wrapper-Methode implementieren, die den Wertebereich überprüft. |

Aufgabe 1.1b)

* Frage 2: Wie kann die Objekterzeugung mit Hilfe einer zusätzlichen Klasse durchgeführt werden? In welchem Package sollte diese zusätzliche Klasse liegen?

|  |
| --- |
| Es kann eine zusätzliche Klasse (z.B. Factory) mit einer statischen Methode erstellt werden, welche die Objekterzeugung übernimmt. Es sollte ein neues Package für die Factory-Klass erstellt werden. |

* Frage 3: Welches Entwurfsmuster liegt für diesen Anwendungsfall nahe? Welchen Vorteil bringt die Nutzung dieses Entwurfsmusters?

|  |
| --- |
| Es handelt sich um das Factory-Pattern, das hat z.B. den Vorteil, dass die Objekterstellung zentral gesteuert werden kann. Außerdem können z.B. die Anzahl erstellter Objekte kontrolliert werden. |

Aufgabe 1.1c)

* Frage 4: Warum sollten Testfälle in einer separaten Test-Klasse implementiert werden?

|  |
| --- |
| Damit sie z.B. klar getrennt vom Production Code liegen, nicht versehentlich geliefert werden und separat gebaut werden können. |

* Frage 5: Wozu dienen die Äquivalenzklassen im Blackbox-Test?

|  |
| --- |
| Um sich einen Überblick darüber zu verschaffen, welche Inputs abgefangen werden müssen und sicherzustellen, dass alle Testfälle abgedeckt sind. |

* Frage 6: Warum lässt sich für die Klasse Client nicht ohne weiteres ein Blackbox-Test umsetzen?

|  |
| --- |
| “void“ Methoden sind schwierig zu testen, da sie keine kategorisierbaren Rückgabewerte für die Assertions liefern. |

Gruppenmitglieder:   
Winkelholz, Patrick  
Braun, Martin  
Knöpfel, Joshua  
Bucher, Fabian