1. Vorbemerkungen

In dieser Aufgabe soll eine einfache objektorientierte Middleware konzipiert und in Java realisiert werden, mit deren Hilfe Methodenaufrufe eines entfernten Objektes möglich sind. Die Middleware soll dabei auch nebenläufige Aufrufe von Methoden - auch desselben Objektes - unterstützen.

Applikationen, die die Middleware benutzen werden, existieren bereits. Deren Funktionalität ist für die Middleware nicht relevant; deshalb werden die Schnittstellen nur formal definiert.

2. Middleware und Schnittstellen

Die Middleware soll auf der obersten Ebenen in vier Packages aufgeteilt werden:

- Package **name_service** mit dem Namensdienst, der auf einem gesonderten Rechner läuft und Namen auf Objektreferenzen abbildet. Der Port muss einstellbar sein. (Parameter)
- Package **mware_lib** mit den allgemein für den Betrieb der Middleware benötigten Klassen, unabhängig von der Art der Applikation.
- Package **bank_access**
- Package **cash_access**

Bei den beiden letzteren Packages handelt es sich um applikationsspezifische Schnittstellen der entfernt aufrufbaren Methoden der Applikationen.

Zulässige Abhängigkeiten:

Extern nur von JRE; innerhalb der Middleware dürfen nur *bank_access* und *cash_access* von *mware_lib* abhängen. Eine Applikation bindet stets das Package *mware_lib* ein, sowie fallweise noch *bank_access* oder/und *cash_access*. Der Namensdienst ist vollständig in *name_service*.

In den genannten vier Packages muss die gesamte Middleware enthalten sein. Die Packages *mware_lib*, *bank_access* und *cash_access* dürfen zusammen nicht grösser als 3 MByte sein.

Schnittstellendefinitionen:

Die nachfolgenden Klassen müssen im Package auf der obersten Ebenen definiert sein:

Package **mware_lib***:* (Schnittstellen zu den Diensten der Middleware)

```
public class ObjectBroker {  //- Front-End der Middleware -
       public static ObjectBroker init(String serviceHost,
                                 int listenPort, boolean debug) { ... }
          // Das hier zurückgelieferte Objekt soll der zentrale Einstiegspunkt
          // der Middleware aus Anwendersicht sein.
          // Parameter: Host und Port, bei dem die Dienste (Namensdienst)
          //            kontaktiert werden sollen. Mit debug sollen Test-
          //            ausgaben der Middleware ein- oder ausgeschaltet werden
          //            können.

       public NameService getNameService() {...}
          // Liefert den Namensdienst (Stellvetreterobjekt).

       public void shutDown() {...}
          // Beendet die Benutzung der Middleware in dieser Anwendung.
}

public abstract class NameService {     //- Schnittstelle zum Namensdienstes -
```

```
        public abstract void rebind(Object servant, String name);
           // Meldet ein Objekt (servant) beim Namensdienst an.
           // Eine eventuell schon vorhandene Objektreferenz gleichen Namens
           // soll überschrieben werden.

        public abstract Object resolve(String name);
           // Liefert eine generische Objektreferenz zu einem Namen. (vgl. unten)
}
```

Verwendungsbeispiel in einer Serverapplikation:

```
      ...
ObjectBroker objBroker = ObjectBroker.init(host, port, false);
NameService nameSvc = objBroker.getNameService();
nameSvc.rebind((Object) konto, KontoID);
   ...
objBroker.shutDown();
```

Verwendungsbeispiel in einer Clientapplikation:

```
   ...
ObjectBroker objBroker = ObjectBroker.init(host, port, false);
NameService nameSvc = objBroker.getNameService();
Object rawObjRef = nameSvc.resolve(KontoID);      //generische Referenz
AccountImplBase konto = AccountImplBase.narrowCast(rawObjRef);
                          //liefert spezialisiertes Stellvertreterobjekt
   ...
double b = konto.getBalance();
   ...
objBroker.shutDown();
```

Package **bank_access** (applikationsspezifische Schnittstelle – formale Definition):

```
public abstract class ManagerImplBase {
       public abstract String createAccount(String owner, String branch)
                           throws InvalidParamException;
       public static ManagerImplBase narrowCast(Object rawObjectRef) {...}
}

public abstract class AccountImplBase {
       public abstract void transfer(double amount) throws OverdraftException;
       public abstract double getBalance();
       public static AccountImplBase narrowCast(Object rawObjectRef) {...}
}

public class InvalidParamException extends Exception {
       public InvalidParamException(String message) { super(message);}
}
public class OverdraftException extends Exception {
       public OverdraftException(String message) { super(message);}
}
```

Package **cash_access** (applikationsspezifische Schnittstelle – formale Definition):

```
public abstract class TransactionImplBase {
       public abstract void deposit(String accountID, double amount)
                           throws InvalidParamException;
       public abstract void withdraw(String accountID, double amount)
                           throws InvalidParamException, OverdraftException;
       public abstract double getBalance(String accountID)
                           throws InvalidParamException;
       public static TransactionImplBase narrowCast(Object rawObjectRef) {...}
}
```

```
public class InvalidParamException extends Exception {
       public InvalidParamException(String message) { super(message);}
}
public class OverdraftException extends Exception {
       public OverdraftException(String message) { super(message);}
}
```

Anmerkungen: Die Middlware kann in den **ImplBase*-Klassen weitere Methoden und Interfaces implementieren. Die Anwendungen kennen und benutzen aber nur die oben definierten Klassen und Methoden. Die Methode `narrowCast` soll ein Stellvertreterobjekt zu einer generischen Objektreferenz liefern und ist jeweils von der Middleware zu implementieren.

Fehlerbehandlung:

Tritt beim Aufruf einer entfernten Methode eine der o.g. Exceptions auf, ist diese von der Middleware in Typ und Meldungstext unverändert an die aufrufende Applikation weiterzuleiten - so, als wäre sie dort lokal geworfen worden. In anderen Fehlerfällen soll eine geeignete Exception mit aussagefähiger Meldung geworfen werden.

Stringparameter können die Zeichen 'A'...'Z', 'a'...'z', '0'...'9', '.', '-' und *Blank* enthalten. Auch der Wert `null` ist zulässig.

3. Hinweise und Tipps:

- Überlegen Sie sich, welche Informationen für einen Aufruf ausgetauscht werden müssen. Entwerfen Sie ein geeignetes Request/Reply-Protokoll.
- Sockets sollen nur in möglichst wenigen Klassen verwendet werden.
- Alle beteiligten Komponenten müssen auf separaten Rechnern im Labor unter Linux laufen können.

Vorbereitung: Bis zum **Sonntagabend 20:00 Uhr** vor dem Praktikumstermin ist ein Konzeptpapier als **PDF**-Dokument (mit ausgefülltem Dokumentationskopf) per E-Mail über den Abgabeverteiler abzugeben. Darin ist die Architektur der Middleware und deren interne Kommunikation mit geeigneten Diagrammen zu dokumentieren.

Bis zum Praktikumstermin ist der Entwurf ggf. soweit zu verfeinern, dass die Implementierung im Praktikum abgeschlossen werden kann.

Die **Vorführung** beginnt um ca. **10:50**. Jedes Team bekommt einen Satz Applikationen, mit dem ein Testlauf mit der eigenen Middleware durchzuführen ist. (Die von der Hauptseite herunterladbaren Applikationen sind hier nicht zu verwenden!)

Abgabe als ZIP-Archiv **am Ende des Praktikums**. Ist eine Überarbeitung nötig, wird diese bis spätestens Donnerstag 20:00 Uhr nach dem Praktikum akzeptiert. Gibt es auch dann noch Beanstandungen, gilt die Aufgabe als nicht erfolgreich bearbeitet.

Der Inhalt des Archivs muss wie folgt strukturiert sein:

- README-Datei, die angibt, wie der Namensdienst von der Kommandozeile zu starten ist.
- Binärcode (bin/) mit den vier Middleware-Packages (Verzeichnisse, keine *.jar)
- Quellcode (src/),

- Logs der Testläufe aus der Vorführung,
- aktualisierter Dokumentationskopf.

Abgaben, die diese Form nicht erfüllen oder unvollständig sind, werden nicht akzeptiert.

Voraussetzung für die erfolgreiche Bearbeitung ist eine fristgerecht eingereichte, ausreichende Vorbereitung. Die Befragung und die Vorführung müssen erfolgreich absolviert werden. Im Übrigen gelten die aus den vorangegangen Aufgaben bekannten Regelungen.