Übung 11 – Aufgabe 1

Muster 1: Java.io : DataInput

Die Strategie bietet Algorithmen um Bytes aus einem binären Datenstrom zu lesen und aus Ihnen Daten in einem der primitiven Typen aus Java zu rekonstruieren.

Klassen:

DataInputStream, FileCacheImageInputStream, FileCacheImageOutputStream, FileImageInputStream, FileImageOutputStream, ImageInputStreamImpl, ImageOutputStreamImpl, MemoryCacheImageInputStream, MemoryCacheImageOutputStream, ObjectInputStream, RandomAccessFile

Methoden:

readBoolean() – Liest einen Input in Form eines Bytes und gibt true zurück, wenn dieses nicht 0 ist, andernfalls false

readByte() - Liest ein Eingabe-Byte ein und gibt dieses zurück

readChar() – Bekommt als Parameter zwei Bytes und gibt als Ergebnis einen Char zurück

readDouble() – Bekommt als Parameter 8 Bytes und gibt einen Double zurück

readFloat() - Bekommt als Parameter 4 Bytes und gibt einen Float zurück

readFully(byte[] b) – Bekommt als Eingabe einige Bytes und speichert diese in dem Array b

readFully(byte[] b, int off, int len) – Liest “len” Bytes von einem Eingabe-Stream

readInt()- Bekommt als Parameter 4 Bytes und gibt einen Integer zurück

readLine() – Liest die nächste Zeile vom Eingabe-Stream

readLong() – Bekommt als Parameter 8 Bytes und gibt einen Long zurück

readShort() - Bekommt als Parameter 2 Bytes und gibt einen Short zurück

readUnsignedByte() – Liest einen Byte ein, castet ihn in einen Integer und gibt ihn als Ergebnis zurück. Dieses ist im Bereich von 0 – 255.

readUnsignedShort() - Bekommt als Parameter 2 Bytes und gibt einen Int im Bereich von 0 – 65535 zurück

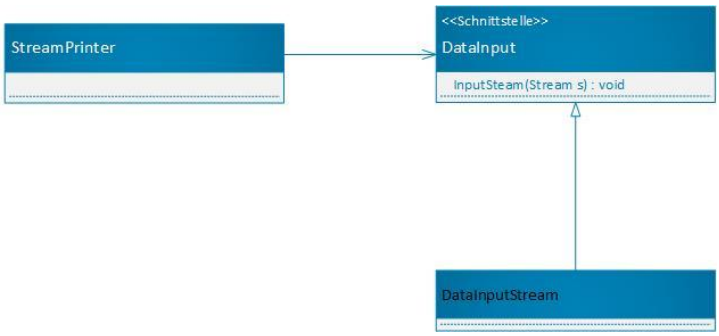
readUTF() – Liest einen String ein, welcher in einem modifizierten UTF-8-Format verschlüsselt wurde

skipBytes(int n) – Versucht die nächsten n Bytes im Eingabe-Stream zu überspringen. Die übersprungenen Bytes werden dabei verworfen

```
public class StreamPrinter implements DataInput{

        public void setInputStream(DataInput stream){
                InputStr = stream;
        }

        public void print(){
                DataInput InputStr = this.InputStr;
                System.out.println(InputStr.readLine());
        }
}
```



Client Code:

```
public class testClass{

        StreamPrinter s = new StreamPrinter();
        s.setInputStream(new DataInputStream());

}
```

Muster 2: Java.io : DataOutput

Die Strategie bietet Algorithmen um aus den primitiven Datentypen in Java eine Folge von Bytes zu generieren und diese in einen binären Output-Stream zu schreiben.

Klassen:

DataOutputStream, FileCacheImageOutputStream, FileImageOutputStream, ImageOutputStreamImpl, MemoryCacheImageOutputStream, ObjectOutputStream, RandomAccessFile

Methoden:

write(byte[] b) – Schreibt alle Bytes im Array b auf den Output-Stream

write(byte[] b, int off, int len) – Schreibt “len” Bytes aus dem Array b geordnet auf den Output-Stream

write(int b) – Schreibt die 8 Bits mit der niedrigsten Ordnung des Argumentes b auf den Output-Stream

writeBoolean(boolean v) – Schreibt einen Boolean auf den Output-Stream

writeByte(int v) - Schreibt die 8 Bits mit der niedrigsten Ordnung des Argumentes b auf den Output-Stream

writeBytes(String s) – Schreibt einen String auf den Output-Stream

writeChar(int v) – Schreibt einen Char, welcher aus zwei Bytes besteht auf den Output-Stream

writeChars(String s) – Schreibt jeden Char des Strings s auf den Output-Stream. Die Reihenfolge ist vorgegeben und es werden für jeden Char zwei Bytes veranschlagt.

writeDouble(double v) – Schreibt einen Double, welcher aus 8 Bytes besteht auf den Output-Stream.

writeFloat(float v) – Schreibt einen Float, welcher aus 4 Bytes besteht auf den Output-Stream

writeInt(int v) - Schreibt einen Int, welcher aus 4 Bytes besteht auf den Output-Stream

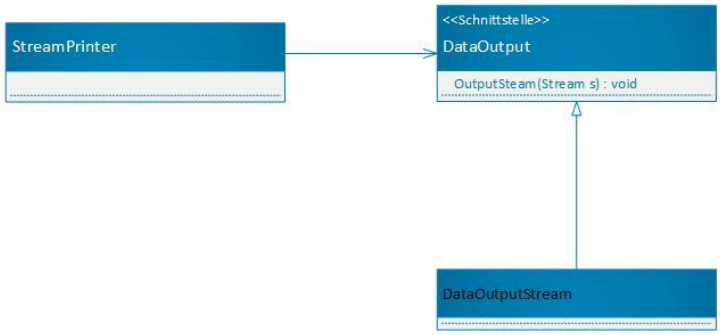
writeLong(long v) - Schreibt einen Long, welcher aus 8 Bytes besteht auf den Output-Stream.

writeShort(int v) – Schreibt zwei Bytes auf den Output-Stream um den Wert des Arguments zu repräsentieren

```
public class StreamPrinter implements DataOutput{

        public void setOutputStream(DataOutput stream){
                OutputStr = stream;
        }

        public void print(){
                DataOutput OutputStr = this.OutputStr;
                OutputStr.write(„hallo“);
        }
}
```



Client Code:

```
public class testClass{

        StreamPrinter s = new StreamPrinter();
        s.setOutputStream(new DataOutputStream());

}
```

Muster 3: Java.lang : Comparable<T>

Mit Hilfe dieser Strategie, kann eine Ordnung von beliebigen Objekten festgelegt werden. Dabei kann diese Ordnung für jede Klasse , die dieses Interface implementiert anders sein.

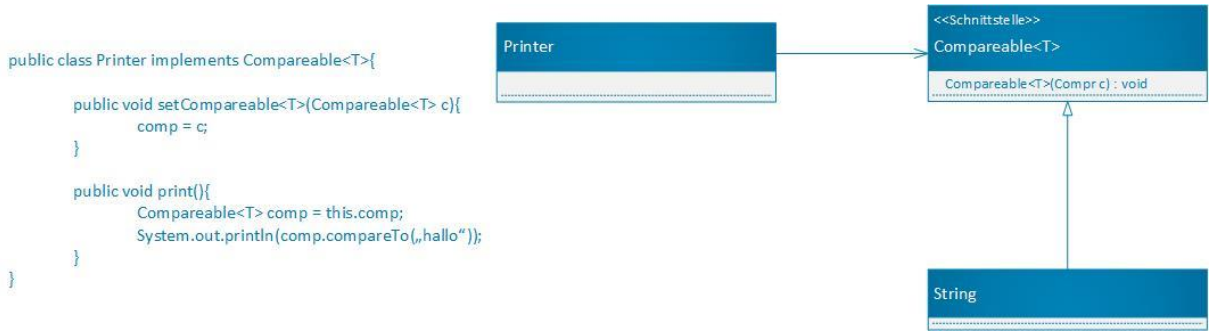
Klassen:

AbstractRegionPainter.PaintContext.CacheMode, AccessMode, AclEntryFlag, AclEntryPermission, AclEntryType, AddressingFeature.Responses, Authenticator.RequestorType, BigDecimal, BigInteger, Boolean, Byte, ByteBuffer, Calendar, CertPathValidatorException.BasicReason, Character, Character.UnicodeScript, CharBuffer, Charset, ClientInfoStatus, CollationKey, Component.BaselineResizeBehavior, CompositeName, CompoundName, CRLReason, CryptoPrimitive, Date, Date, Desktop.Action, Diagnostic.Kind, Dialog.ModalExclusionType, Dialog.ModalityType, Double, DoubleBuffer, DropMode, ElementKind, ElementType, Enum, File, FileTime, FileVisitOption, FileVisitResult, Float, FloatBuffer, Formatter.BigDecimalLayoutForm, FormSubmitEvent.MethodType, GraphicsDevice.WindowTranslucency, GregorianCalendar,

GroupLayout.Alignment, IntBuffer, Integer, JavaFileObject.Kind, JTable.PrintMode, KeyRep.Type, LayoutStyle.ComponentPlacement, LdapName, LinkOption, Locale.Category, Long, LongBuffer, MappedByteBuffer, MemoryType, MessageContext.Scope, Modifier, MultipleGradientPaint.ColorSpaceType, MultipleGradientPaint.CycleMethod, NestingKind, Normalizer.Form, NumericShaper.Range, ObjectName, ObjectStreamField, PKIXReason, PosixFilePermission, ProcessBuilder.Redirect.Type, Proxy.Type, PseudoColumnUsage, Rdn, Resource.AuthenticationType, RetentionPolicy, RoundingMode, RowFilter.ComparisonType, RowIdLifetime, RowSorterEvent.Type, Service.Mode, Short, ShortBuffer, SOAPBinding.ParameterStyle, SOAPBinding.Style, SOAPBinding.Use, SortOrder, SourceVersion, SSLEngineResult.HandshakeStatus, SSLEngineResult.Status, StandardCopyOption, StandardLocation, StandardOpenOption, StandardProtocolFamily, String, SwingWorker.StateValue, Thread.State, Time, Timestamp, TimeUnit, TrayIcon.MessageType, TypeKind, URI, UUID, WebParam.Mode, Window.Type, XmlAccessOrder, XmlAccessType, XmlNsForm

Methoden:

compareTo(T o) – Vergleicht dieses Objekt mit dem angegebenen und gibt als Ergebnis einen Int zurück, welcher die Rangordnung zwischen den beiden spezifiziert.



Client Code:

```
public class testClass <T>{

        Printer p = new Printer();
        p.setCompareable<T>(new Compareable<T>());

}
```

Muster 4: Java.lang : Iterable<T>

Mit Hilfe dieser Strategie kann über verschiedene Typen, bzw. Objekte iteriert werden (Je nach endgültiger Implementierung).

Klassen:

AbstractCollection, AbstractList, AbstractQueue, AbstractSequentialList, AbstractSet, ArrayBlockingQueue, ArrayDeque, ArrayList, AttributeList, BatchUpdateException, BeanContextServicesSupport, BeanContextSupport, ConcurrentLinkedDeque, ConcurrentLinkedQueue, ConcurrentSkipListSet, CopyOnWriteArrayList, CopyOnWriteArraySet, DataTruncation, DelayQueue, EnumSet, HashSet, JobStateReasons, LinkedBlockingDeque, LinkedBlockingQueue, LinkedHashSet, LinkedList, LinkedTransferQueue, PriorityBlockingQueue, PriorityQueue, RoleList, RoleUnresolvedList, RowSetWarning, SerialException, ServiceLoader, SQLClientInfoException, SQLDataException, SQLException, SQLFeatureNotSupportedException, SQLIntegrityConstraintViolationException, SQLInvalidAuthorizationSpecException, SQLNonTransientConnectionException, SQLNonTransientException, SQLRecoverableException, SQLSyntaxErrorException, SQLTimeoutException, SQLTransactionRollbackException, SQLTransientConnectionException, SQLTransientException, SQLWarning, Stack, SyncFactoryException, SynchronousQueue, SyncProviderException, TreeSet, Vector

Methoden:

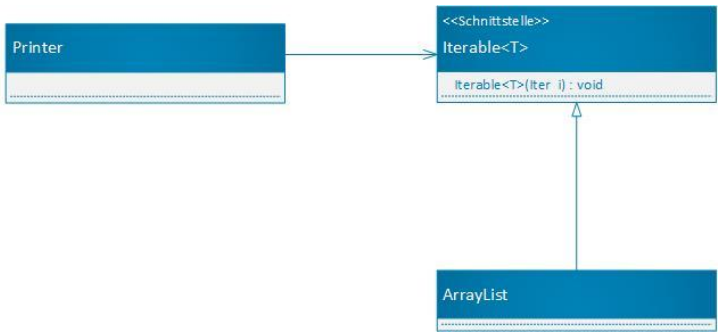
iterator() – Gibt einen Iterator über ein Set von Elementen des Typs T zurück.

Returns an iterator over a set of elements of type T.

```
public class Printer implements Iterable<T>{

        public void setIterable<T>(Iterable<T> i){
                iter = i
        }

        public void print(){
                Iterable<T> iter = this.iter;
                System.out.println(„ups, schlechtes beispiel“);
        }
}
```



Client Code:

```
public class testClass <T>{

        Printer p = new Printer();
        p.setIterable<T>(new Iterable<T>());

}
```

Muster 5: Java.lang : Readable

Durch diese Strategie wird eine lesbare Quelle spezifiziert.

Klassen:

BufferedReader, CharArrayReader, CharBuffer, FileReader, FilterReader, InputStreamReader, LineNumberReader, PipedReader, PushbackReader, Reader, StringReader

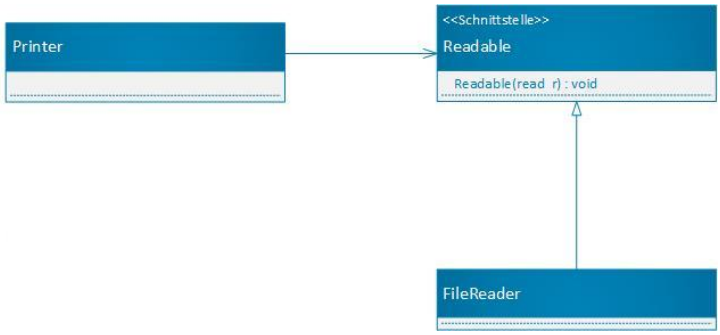
Methoden:

read(CharBuffer cb) - Versuche Zeichen in dem angegebenen Char-Buffer zu lesen.

```
public class Printer implements Readable{

        public void setReadable<T>(Readable r){
                read = r
        }

        public void print(){
                Readable read = this.read;
                System.out.println(read.read(new File(„/test“));
        }
}
```



Client Code:

```
public class testClass {

        Printer p = new Printer();
        p.setReadable(new Readable(File f));

}
```