```java
package hyperDap.base.helpers;

import java.math.BigDecimal;
import java.math.BigInteger;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.LocalTime;
import java.time.MonthDay;
import java.time.OffsetDateTime;
import java.time.OffsetTime;
import java.time.Year;
import java.time.YearMonth;
import java.time.ZoneId;
import java.time.ZoneOffset;
import java.time.ZonedDateTime;
import java.util.HashMap;
import java.util.function.Function;

/**
 * A class used to parse objects to the desired types. This may be achieved simply by casting
 * correctly between different classes, ensuring that the final Object is otherwise as identical to
 * the original as possible, or by parsing values to produce a desired type.
 *
 * @author soenk
 *
 */

public final class Parser {

  private Parser() {}

  /**
   * A map of parseable classes to their correct parsing functions. Used by
   * {@link #parse(String, Class)}.
   */
  private static HashMap<Class<?>, Function<String, ?>> parseMap = new HashMap<>();
  static {
    parseMap.put(boolean.class, Boolean::parseBoolean);
    parseMap.put(byte.class, Byte::parseByte);
    parseMap.put(short.class, Short::parseShort);
    parseMap.put(int.class, Integer::parseInt);
    parseMap.put(long.class, Long::parseLong);
    parseMap.put(double.class, Double::parseDouble);
    parseMap.put(float.class, Float::parseFloat);
    parseMap.put(Boolean.class, Boolean::valueOf);
    parseMap.put(Byte.class, Byte::valueOf);
    parseMap.put(Short.class, Short::valueOf);
    parseMap.put(Integer.class, Integer::valueOf);
    parseMap.put(Long.class, Long::valueOf);
    parseMap.put(Double.class, Double::valueOf);
    parseMap.put(Float.class, Float::valueOf);
    parseMap.put(String.class, String::valueOf);
    parseMap.put(BigDecimal.class, BigDecimal::new);
    parseMap.put(BigInteger.class, BigInteger::new);
    parseMap.put(LocalDate.class, LocalDate::parse);
    parseMap.put(LocalDateTime.class, LocalDateTime::parse);
    parseMap.put(LocalTime.class, LocalTime::parse);
    parseMap.put(MonthDay.class, MonthDay::parse);
    parseMap.put(OffsetDateTime.class, OffsetDateTime::parse);
    parseMap.put(OffsetTime.class, OffsetTime::parse);
    parseMap.put(Year.class, Year::parse);
    parseMap.put(YearMonth.class, YearMonth::parse);
    parseMap.put(ZonedDateTime.class, ZonedDateTime::parse);
    parseMap.put(ZoneId.class, ZoneId::of);
    parseMap.put(ZoneOffset.class, ZoneOffset::of);
  }

  /**
   * Take a String representation of the desired Object and parse it to the desired {@link Class}
   * representation.
```

```
71      * <p>
72      * Used as: {@code parse("123.45",Double.class)} and can be used on primitive types.
73      * <p>
74      * Taken from <a href="https://ideone.com/WtNDN2">this post</a> and referenced in
        <a href=
75      *
        "https://stackoverflow.com/questions/36368235/java-get-valueof-for-generic-subclass
        -of-java-lang-number-or-primitive">this
76      * Stackoverflow discussion</a>.
77      *
78      * @param stringRepresentation
79      * @param classReference
80      * @return
81      */
82     @SuppressWarnings({"rawtypes", "unchecked"})
83     public static Object parse(String stringRepresentation, Class classReference) {
84       Function<String, ?> function = parseMap.get(classReference);
85       if (function != null)
86         return function.apply(stringRepresentation);
87       if (classReference.isEnum())
88         return Enum.valueOf(classReference, stringRepresentation);
89       throw new UnsupportedOperationException(
90           String.format("Parsing String '%s' to class '%s' has failed",
             stringRepresentation,
91               classReference.getName()));
92     }
93
94   }
95
```