DAC 0521 MET-N Writeup - Java Module 1 Session 10

Primitive Value Type	Wrapper Class Type
boolean	java.lang.Boolean
char	java.lang.Character
byte	java.lang.Byte
short	java.lang.Short
int	java.lang.Integer
long	java.lang.Long
float	java.lang.Float
double	java.lang.Double

Boxing - Enclosing a primitive-type value within an instance of its wrapper class so that it can be used as a reference type is called boxing.
In Java (5.0 onwards) the compiler automatically inserts the call (to valueOf method of wrapper class) to perform this operation in conversion of a primitive type into its wrapper class type.

Unboxing - Extracting a primitive-type value from the instance of its wrapper class is called unboxing.
In Java (5.0 onwards) the compiler automatically inserts the call (to primitiveValue method of wrapper class) to perform this operation in conversion of a primitive type from its wrapper class type.

Java Generics - It is syntactical support offered by Java (5.0 onward) language for implementing recurring code patterns which can be reused with different reference types in a type-safe manner. It enables the Java compiler to identify matching reference types and perform explicit conversions for those types.
A generic declaration (method, class, interface) in Java contains at least one type argument which can be substituted by any reference type at compile time with support for
(1) Type Erasure - The type argument is always replaced by java.lang.Object at runtime and as such at compile time it only supports methods of java.lang.Object by default.
(2) Bounded Type - The type argument T can appear in a declaration as <T extends U> to indicate that it can only be substituted by a reference type which supports implicit conversion to (inherits from) type U and as such it also supports members of U
(3) Wildcard Substitution - A generic type G with type argument T can appear in a

declaration as:

(a) G<? extends U> which can be substituted by G<V> where V is a reference type which supports implicit conversion to U but only members in which T does not appear as an argument type can be applied to this declaration. Also G<?> is equivalent to G<? extends java.lang.Object>

(b) G<? super U> which can be substituted by G<V> where V is a reference type which supports implicit conversion from U but only members in which T does not appear as return type can be applied to this declaration.