Datatypes – Character Class

The following methods of conversion are available in the Character class …

To convert from the Class datatype to the primitive …

*char char2 = newchar.**charValue**();*

To convert the case of the char ...

```
char char2 = newchar2.charValue();

char char3 = Character.toLowerCase(char2);
char char4 = Character.toUpperCase(char2);
char char5 = Character.toTitleCase(char2);
String string1 = Character.toString(char2);
```

Datatypes – Casting and Converting

Occasionally the Developer will need to change variables to different datatypes, this is called *Casting* … Not all datatypes will cast into other datatypes, for example a boolean value will not cast into or cannot be cast from, the following slide is a representation of datatypes and their casting with other datatypes.

The basic syntax of casting is as follows, the receiving variable is on the left, the datatype of the receiving variable is in brackets and the value or variable which is to be cast is on the extreme right ...

receivingvar = (char)castingvar;

This is called an explicit cast, however some datatypes will 'cast' implicitly by just using an equals sign ...

Datatypes – Casting and Converting

	boolean	byte	char	double	float	int	long	short	String
boolean		✍	✍	✍	✍	✍	✍	✍	✍
byte	✍		(✍)	✍	✍	(✍)	✍	(✍)	✍
char	✍	(✍)		(✍)	(✍)	(✍)	(✍)	(✍)	✍
double	✍	✍	(✍)		✍	✍	✍	✍	✍
float	✍	✍	(✍)	✍		✍	✍	✍	✍
int	✍	(✍)	(✍)	✍	✍		(✍)	(✍)	✍
long	✍	✍	(✍)	✍	✍	(✍)		✍	✍
short	✍	(✍)	(✍)	✍	✍	(✍)	✍		✍
String	✍	✍	✍	✍	✍	✍	✍	✍	

Ticks in brackets mean that the casting must take place explicitly, casting datatypes without brackets isnt necessary but not a compilation error.

Datatypes – Casting and Converting

This is an example of casting a int from a byte ...

```
byte b = 12;
int  i = 0;
i = (int)b;
```

When casting from int to char, an ascii conversion takes place, and vice versa int to ascii character ..

```
char c;
int  ci = 65;
c = (char)ci;
System.out.println(ci);
```

A

Datatypes – Casting and Converting

Casting from a boolean or to a boolean isnt possible and given it only has two states it doesn't take much to program around the problem, if the states need to be saved in a different datatype, not being able to cast from or to a String could be a problem, however a series of methods have been supplied which get round this problem.

The method is called *String.valueOf* and it will convert any of the mentioned datatypes into a String (even booleans !) and in addition will convert arrays of chars (these will be covered later in the Course).