**형변환**

|  |  |
| --- | --- |
| 1 | 아래의 코드에 대한 설명 중 틀린 것을 모두 고르면?  **public** **class** VarTest2 {  **public** **static** **void** main(String[] args) {  **short** s1=3; (가)  **short** s2=4; (나)    **int** a1 = 3;(다)  **int** a2 = 4;(라)  **short** sum = s1+s2; (마)  **int** result = a1+a2;(바)  }  }  (1) (가)와 (나) 는 메모리 공간을 2byte씩 차지한다.  (2) (다)와 (라)는 메모리 공간을 4byte씩 차지한다.  ~~(3) 데이터의 크기가 3과 4와 같이 크기가 아주 작다면 int 형보다는 short 형을 사용하는 편이 더 효율적~~  ~~이므로, 개발자는 연산 처리하려는 데이터의 크기를 언제나 민감하게 체크해야 한다.~~  **연산 수행 시 int로 캐스팅되기에 별로 이득이 아님. 단, 연산을 하지 않는 경우, 메모리 이득을 위하여 short등을 사용하는 것은 이득이 되는 부분임. (메모리 로드 등)**  ~~(4) (마)는 7의 결과가 sum 변수에 대입되므로, 에러가 없는 코드이다~~  (5) 자바의 컴파일러는 정수 자료형의 연산에 대해서는 32비트로 처리하므로, 개발자가 int 형 이하의  자료형(byte, short)을 대상으로 연산을 수행할 경우, 자동으로 int 형으로 변환되어 연산이 수행되어  진다  (6) ~~int 형보다 작은 자료형간의~~ **~~연산시~~** ~~발생하는 int형으로의 자동 형변환 현상은 64bit 기반의 컴퓨터 일~~  ~~경우 발생하지 않는다~~  **JAVA는 32비트에 최적화되어 있기에 하드웨어를 따라가지 않는다. (과거 버전과의 호환이 중요)**  (7) long 형 들 간의 연산 수행시에도 자동 int형으로의 형변환이 적용된다 |
| 2 | 다음 설명 중 틀린 것을 모두 고르면?  (1) int형 보다 작은 정수형은 int형으로 자동 형변환되어 연산이 수행되어 진다.  (2) ~~(1)과 같은 이유로 byte, short 형은 자바에서 사용되어질 일이 없는 쓸데없는 자료형에 불과하다.~~  **버퍼 프로그래밍 등에 이용**  (3) ~~float x = 3.14; 라고 선언시 3.14는 실수이므로 정상적으로 컴파일 될 것이다.~~  (4) **자바에서 실수는 무조건 double 형으로 연산된다.** |
| 3 | 다음 설명 중 맞는 것은?  **public** **class** VarTest1 {  **public** **static** **void** main(String[] args) {  **boolean** b=**true**;  **int** x=3;  **short** y=9;  **char** c='K';    **int** r1= b+x; //(가)  **short** r2 = y+c; //(나)  **int** r3=x+c; //(다)  **boolean** r4=b+c; //(라)  }  }  (1) ~~(가)에서 b는 int 형으로 자동 변환되어 연산이 수행되므로 정상 컴파일 될 것이다.~~  (2) ~~(나)에서 x는 정수고 c는 문자이므로 연산수행이 불가능하며 컴파일 에러가 발생할 것이다.~~  (3) (다)에서 c는 int로 형변환되므로 정상 컴파일 될 것이다.  (4) ~~(라)에서 c는 boolean 형으로 변환되므로 정상 컴파일 될 것이다.~~ |
| 4 | 다음 설명 중 틀린 것은?  (1) 작은 자료형의 데이터가 큰 자료형의 데이터로 변환되는 모습을 **promotion** 이라고 한다.  (2) 큰 자료형으로 선언된 데이터가 작은 자료형의 데이터로 변환되는 모습을 **demotion** 이라 한다.  (3) ~~boolean 형과 char 형간에는 형변환이 가능하다.~~  (4) 아래처럼 작은 자료형으로 선언된 데이터를 큰 자료형으로 받을 경우 , 데이터의 손실이 발생하지  않는다  byte b =3;  short s = b;  (5) ~~아래와 같이 a를 c에 대입할때, c가 용량이 더 크므로 변환하는데 아무런 문제가 없다~~  **음수의 존재 때문에 (-128~127), 범위가 초과된다.**  **byte** a=9;  **char** c=a; |
| 5 | 다음 중 맞는 것을 모두 고르면?  (1) 아래의 문장은 올바르게 컴파일 될 것이다.  byte b=3;  int a=7;  a=b;  (2) ~~아래의 문장은 올바르게 컴파일 될 것이다.~~  ~~short s=7;~~  ~~byte b=3;~~  ~~b=s;~~  (3)아래의 문장은 올바르게 컴파일 될 것이다.  char a=3;  int b=5;  b=a;  (4) 아래의 문장은 올바르게 컴파일 될 것이다.  int a=5;  char c=65;  a=c; |
| 6 | 아래의 코드에 대한 설명 중 틀린 것을 고르면?  short a = 3;  short b = 9;  short c = a + b;  (1) ~~위 코드는 short 형들 간 연산을 수행한 후, 그 결과를 short에 담았으므로 아무런 문제가 없이 컴파일~~  ~~될 것이다.~~  (2) 일반적으로 데이터의 연산 수행 시 피연산자들 간의 데이터형이 같아야 CPU가 연산을 수행한다.  (3) 위 코드에서 a + b의 덧셈 연산을 수행하기 전에 a와 b 는 먼저 int 형으로 형이 변환되어 진다  (4) ~~연산 수행 전 int형으로 변환되는 현상은 자바의 모든 숫자형 byte, short, int, long, float, double 에~~  ~~적용되는 원칙이다.~~ |
| 7 | 아래의 코드에 대한 설명 중 틀린 것은?  byte a = 8;  long b = 2;  int c = a + b; (가)  **(1) ~~위 코드의 (가) 부분에서 a는 일단 +연산의 수행 전 int 형으로 변경된다.~~**  **Long 형변환이 일어난다.**  (2) ~~위 코드의(가) 부분에서 b도 일단 연산수행 전에 int 형으로 변경된다.~~  (3) (가) 연산이 수행되고 나면 a+b의 결과값은 long 형이므로 , int c 로 결과를 받을 수 없다.  (4) (가) 부분에서 발생되는 에러를 해결하기 위해서는 c 를 long 형으로 선언하면 컴파일 에러는  발생되지 않는다. |
| 8 | 다음 중 틀린 것은?  (1) 이미 선언된 자료형을 다른 데이터형으로 강제로 바꾸고자 할 경우 사용되는 형변환을 명시적  형변환이라 하며 이때 cast 연산자를 사용할 수 있다.  (2) ~~대입연산자가 사용된 문장에서는 형변환은 발생하지 않는다~~  (3) 서로 다른 데이터형 간에는 연산이 불가능하므로 어떤 한쪽의 데이터형태로 자동적으로 형 변환이  이루어지는 것을 자동형변환이라 한다.  (4) byte, char, short 은 연산 시 int형으로 변환되어 지므로 자동 형변환에 해당한다. |
| 9 | 다음 중 틀린 것은?  (1) ~~형변환은 자바의 기본 자료형들끼리 가능하므로 아래의 문장은 올바르게 수행된다.~~  ~~boolean k=true;~~  ~~int a=k;~~  (2) int형 보다 작은 자료형들간의 연산 수행시, int 형으로 자동 형변환 되어 연산이 수행된다.  (3) int형 보다 큰 자료형들간의 연산 수행시엔 , int형으로의 형변환은 발생하지 않는다.  (4) char 형은 사실상 숫자형과 마찬가지이므로 연산 수행시 int 형으로의 변환이 발생하며, 아래 문장은  올바르게 수행된다.  int a=65;  char c=2;  c=(char)a; |