**Ojbect & Class II**

1. Method Overloading

|  |
| --- |
| //Overloading 테스트  **class** Boiler{      **void** tempUp(){  temp++;  }  **void** tempUp(**int** amount){  temp = temp + amount;  }  }  **public** **class** OverloadingTest {    **public** **static** **void** main(String[] args) {    System.***out***.println("현재 온도 = " + bo.temp);  bo.tempUp();  System.***out***.println("bo.tempUp() 메소드 호출 후 현재 온도 = " + bo.temp);  bo.tempUp(20);  System.***out***.println("bo.tempUp(20) 메소드 호출 후 현재 온도 = " + bo.temp);  }  } |
| 현재 온도 = 0  bo.tempUp() 메소드 호출 후 현재 온도 = 1  bo.tempUp(20) 메소드 호출 후 현재 온도 = 21 |

2. Static Method & Instance Method

|  |
| --- |
| //Static Method 테스트  **public** **class** StaticMethod {    **void** memMethod1(){  **int** local = memVar;  local = *staticVar*;  *staticMethod1*();  memMethod2();  System.***out***.println("memMethod1");  }  **void** memMethod2(){  System.***out***.println("memMethod2");  }  **static** **void** staticMethod1(){  //int local = memVar;  //memMethod1()  **int** local = st.memVar;  st.memMethod2();  *staticMethod2*();  System.***out***.println("staticMethod1");  }  **static** **void** staticMethod2(){  System.***out***.println("staticMethod2");  }    **public** **static** **void** main(String[] args) {  //StaticMethod.memMethod1();  st.memMethod1();  StaticMethod.*staticMethod2*();  *staticMethod2*();  **int** local = *staticVar*;  //local = memVar;  }  } |
| memMethod2  staticMethod2  staticMethod1  memMethod2  memMethod1  staticMethod2  staticMethod2 |

3. Constructor

|  |
| --- |
| //생성자 테스트  **class** Person{  **int** age;  String name;  **int** height;  **int** weight;      }  **public** **class** ConstructorTest {    **public** **static** **void** main(String[] args) {  Person person1 = **new** Person();  System.***out***.println("person1.name = " + person1.name + ", person1.height = "  + person1.height + ", person1.weight = " + person1.weight +  ", person1.age = " + person1.age);    Person person2 = **new** Person(30);  System.***out***.println("person2.name = " + person2.name + ", person2.height = "  + person2.height + ", person2.weight = " + person2.weight +  ", person2.age = " + person2.age);    Person person3 = **new** Person("goodman");  System.***out***.println("person3.name = " + person3.name + ", person3.height = "  + person3.height + ", person3.weight = " + person3.weight +  ", person1.age = " + person3.age);  }  } |
| person1.name = null, person1.height = 0, person1.weight = 0, person1.age = 0  person2.name = null, person2.height = 0, person2.weight = 0, person2.age = 30  person3.name = goodman, person3.height = 0, person3.weight = 0, person1.age = 0 |

※ 생성자가 발생하는 Exception

생성자도 메소드처럼 Exception을 밖으로 던질수 있으며, check exception일때는 다음과 같이 throws절을 써주어야 한다.

|  |
| --- |
| **package** p04.Method;  **public** **class** Ex\_Account {    **public** Ex\_Account(String accountNo, String ownerName, **int** balance) {    **if**(balance < 0)  **throw** **new** Exception("객체를 생성할수 없습니다.");  **this**.accountNo = accountNo;  **this**.ownerName = ownerName;  **this**.balance = balance;  }  **void** deposit(**int** amount){//예금  balance += amount;  }  //throws Exception: 메소드가 발생 익셉션의 종류를 표시하는 throws절  **int** withdraw(**int** amount) {//인출  **if**(balance < amount)//잔액이 요구금액보다 적으면 Exception발생  ("잔액이 부족합니다.");  // return 0;  balance -= amount;  **return** amount;  }  } |

|  |
| --- |
| **package** p04.Method;  **class** Ex\_Answer {  **public** **static** **void** main(String args[]) {  Ex\_Account obj1 = **new** Ex\_Account("777-777-7777", "연놀부", 10000000);  System.***out***.println(obj1.balance);  Ex\_Account obj2 = **new** Ex\_Account("000-000-000000", "연흥부", -100000);  System.***out***.println(obj2.balance);      } |
| 10000000  객체를 생성할수 없습니다. |

4. this

|  |
| --- |
| //this() 테스트  **class** House{  **int** price;  String dong;  **int** size;  String kind;    **public** House() {  //System.out.println("this보다 앞에서 실행");  **this**(100,32,"상계동","아파트");  }  **public** House(**int** price) {  **this**(price,32,"상계동","아파트");  }  **public** House(**int** price,**int** size) {  **this**(price,size,"상계동","아파트");  }  **public** House(**int** price,**int** size,String dong) {  **this**(price,size,dong,"아파트");  }    }  **public** **class** ThisConstructorTest {  **public** **static** **void** main(String[] args) {    House house2 = **new** House(300);  System.***out***.println("house2.price = " + house2.price + ", house2.size = " +  house2.size + ", house2.dong = " + house2.dong + ", house2.kind = " +  house2.kind);    House house3 = **new** House(300, 40);  System.***out***.println("house3.price = " + house3.price + ", house3.size = " +  house3.size + ", house3.dong = " + house3.dong + ", house3.kind = " +  house3.kind);    House house4 = **new** House(300, 40, "서초동");  System.***out***.println("house4.price = " + house4.price + ", house4.size = " +  house4.size + ", house4.dong = " + house4.dong + ", house4.kind = " +  house4.kind);    House house5 = **new** House(300, 40, "서초동", "빌라");  System.***out***.println("house5.price = " + house5.price + ", house5.size = " +  house5.size + ", house5.dong = " + house5.dong + ", house5.kind = " +  house5.kind);  }  } |
| house1.price = 100, house1.size = 32, house1.dong = 상계동, house1.kind = 아파트  house2.price = 300, house2.size = 32, house2.dong = 상계동, house2.kind = 아파트  house3.price = 300, house3.size = 40, house3.dong = 상계동, house3.kind = 아파트  house4.price = 300, house4.size = 40, house4.dong = 서초동, house4.kind = 아파트  house5.price = 300, house5.size = 40, house5.dong = 서초동, house5.kind = 빌라 |

5. Init Block

|  |
| --- |
| //Initial block test  **class** Good{    }  **public** **class** InitailBlockTest {    **public** **static** **void** main(String[] args) {    Good g1 = **new** Good();  Good g2 = **new** Good();  Good g3 = **new** Good();  System.***out***.println("main");  }  } |
| good의 static 초기화 블록 실행  good의 인스턴스 초기화 블록 실행  good의 인스턴스 초기화 블록 실행  good의 인스턴스 초기화 블록 실행  main |