```
/**
 * 일상생활의객체를추상화하기위한모델링클래스정의
 * 은행계좌객체
 */

class Account {

	//static 초기화블록 :특수한목적의명령어(초기화작업들) 실행
	static{
		System.out.println("초기화블럭실행입니다....1");
		System.out.println("초기화블럭실행입니다....2");
	}
	//클래스변수(static) 선언, 접근제한자붙일수있다.(대체로 public)
	public static final String bankName="하나은행";//실제로할당될때는Account.bankName이렇게할당된다.

	// 인스턴스변수(필드) 선언
	private String accountNum; //계좌번호, 얘네는레퍼런스변수이다. String이객체라서, 얘네는 null로초기화된다.
	private String accountOwner; //예금주
	private intpasswd; //비밀번호
	private long restMoney; //잔고돈

	//디폴트생성자
	public Account(){
		this(null, null); // 얘는밑에생성자호출, 중간꺼는맨아래호출한다.
	}

	//생성자
	public Account(String accountNum, String accountOwner){
		//this.accountNum = accountNum;
		//this.accountOwner = accountOwner;

		//Account(accountNum, accountOwner, 0, 0); 이렇게호출하고싶다. 이럴때는자기자신의생성자를호출할수있다.
		this(accountNum, accountOwner, 0, 0); //임의의값으로넘겨줄수있음
	}

	//this를사용한생성자, 재귀호출이라고도함
	public Account(String accountNum, String accountOwner, intpasswd, long restMoney){
		this.accountNum = accountNum;//여기서 this는인스턴스변수자신을나타낸다.
		this.accountOwner = accountOwner;
		this.passwd = passwd;
		this.restMoney =restMoney;
	}

	//setter/getter 메소드 :외부에서데이터변경할수있도록
```

```
public void setAccountNum(String accountNum){
        this.accountNum    =accountNum;
}

public String getAccountNum(){
        returnthis.accountNum;
}

public void setAccountOwner(String accountOwner){
        this.accountOwner    =accountOwner;
}

public String getAccountOwner(){
        returnthis.accountOwner;
}

public void setPasswd(intpasswd){
        this.passwd = passwd;
}

publicintgetPasswd(){
        returnthis.passwd;
}

public void setRestMoney(long restMoney){
        this.restMoney = restMoney;
}


//인스턴스메소드시작

//입금메소드, 인스턴스변수를모두사용가능(지역변수는함수안에서만)
public long deposit(long money){
        //매개변수는  money
        restMoney += money;//기존의잔고에받은돈넣기
        return restMoney;//값반환
}

//출금메소드
public long withdraw(long money){

        restMoney -= money;//기존의잔고에서돈빼기
        return restMoney;//값반환
}
```

```
	//잔액조회메소드
	public long getRestMoney(){
		returnrestMoney;
	}

	//비밀번호확인메소드
	publicbooleancheckPasswd(int pw){
		return passwd == pw; //같으면 true를반환다르면 false 반환

	}
	//이렇게하면출력시에유연성이증가(데스크탑, 모바일등에구애받지않음)

	public String toString() {

		return getAccountNum()+"\t"+getAccountOwner()+"\t"+"****"+"\t"+getRestMoney();
	}

	@Override
	publicboolean equals(Object obj) {
		booleaneq = false;
		//객체가 Object라서 String등다른객체가들어올경우유효성검사
		if(objinstanceof Account) {
			eq = toString().equals(obj.toString());
		}
		returneq;
		//위임형비교라고한다.
		//return this.toString().equals(obj.toString()) ? true : false;
	}

	//클래스메소드만들기 :인스턴스들이공통으로사용하는메소드
	public static int sum(int a, int b){
		returna+b;
	}
}
```

```
/**
 * 프로그램실행을위한애플리케이션클래스정의
 */

public class AccountExample {
//키워드없으면디렉토리가다른곳에서는사용못함
	public static void main(String[] args) {
		System.out.println("은행계좌애플리케이션시작됨...");

		//클래스로부터객체(인스턴스) 생성방법, 객체를생성해서쓰는거라재사용이가능
		//생성자안만들었을때
		//Account account;//1. 레퍼런스(참조) 변수선언, 4byte가할당되는데값을담기위한변수가아님
		//account = new Account();//2. 인스턴스생성, 할당되는데값이아니라주소로할당이된다.

		//인스턴스의속성과기능사용방법(.을활용해서접근하는데직접접근하는것이아니라주소로접근)
		//account.accountNum = "1111-2222-3333";
		//account.accountOwner= "박소연";
		//account.restMoney= 100000;
		//account.passwd = 1234;

		//생성과동시에초기화할수있음
		Account account;
		account = new Account("1111-2222-3333","박소연",1234, 100000);

		//1. 비밀번호확인 2. 입금 3. 출금하기

		//1. 비밀번호확인
		intpasswd = 1234;
		boolean result = account.checkPasswd(passwd);//메소드호출하면서매개변수주기
		if(result){
			long money = account.deposit(100000);//2. 입금하기
```

```
                System.out.println("입금후잔액 : "+money+" 원");
                money = account.withdraw(10000);//3.출금하기
                System.out.println("출금후잔액 :"+money+" 원");
        }else{
                System.out.println("비밀번호를확인하세요.");
        }

        //System.out.println(account.accountNum);

        //-----------------------------------------------------------------
        //홍길동계좌만들어보기(객체의재사용)
        Account account2 = new Account();

        //지역변수는선언할때초기화안하면에러, 인스턴스변수는자동초기화가된다.
인스턴스안에선언만되어있는상태인데자바가상머신이초기화시킨다.
        //boolean false, 나머지는 0으로초기화된다. String은클래스이름이다. 즉, 객체이다. 그래서 null이라고초기화,
null은아무것도아직참조하고있지않다는뜻
        //인스턴수변수의경우 JVM에의해자동초기화됨.
        //System.out.println(account2.accountNum);
        //System.out.println(account2.accountOwner);
        //System.out.println(account2.restMoney);
        //System.out.println(account2.passwd);


        //set함수를호출해서데이터를넣는다.
        account2.setAccountNum("1111-2222-3333");
        System.out.println(account2.getAccountNum());//get함수를호출해서리턴값을가져온다.
        System.out.println(account2.getAccountOwner());
        System.out.println(account2.getRestMoney());
        System.out.println(account2.getPasswd());

        Account account3 = new Account("2222-3333-4444","홍길동");

        //클래스변수사용하기
        System.out.println(Account.bankName);//인스턴스생성안해도바로접근해서사용할수있다.

        //클래스메소드사용하기
        System.out.println(Account.sum(30, 20));

        //원래는클래스변수변경가능-> final로선언되면안된다.
        //Account.bankName="Hana Bank";
        //System.out.println(Account.bankName);

        //toString테스트
        System.out.println(account);
```

```
                System.out.println("은행계좌애플리케이션종료됨...");
        }


}
```

```
/**
 * 배열을이용한은행계좌관리
 *
 * @author박소연
 *
 */
public class AccountManager {
        private Account[] accounts;//계좌담기위한배열
        private int count=0;//실제담긴계좌갯수

        publicAccountManager() {
                this(50);//기본으로만들어놓는계좌
        }
        publicAccountManager(int size) {
                accounts = new Account[size];
        }
```

```
//getter, setter 메소드
public Account[] getAccounts() {
		return accounts;
}
public void setAccounts(Account[] accounts) {
		this.accounts = accounts;
}
publicintgetCount() {
		return count;
}
public void setCount(int count) {
		this.count = count;
}

/**
 * 계좌개설
 * @paramaccount :계좌정보받아오는객체
 */
public void add(Account account) {

		//은행계좌개설
		accounts[count] = account;
		count++;
}



/**
 * 전체계좌 list 반환
 *
 * @returnlist배열 :실제로계좌등록된것만반환
 */
public Account[] list() {
		Account[] list = new Account[count];
		for (int i = 0; i < count; i++) {
			list[i]= accounts[i];
		}
		return list;
}

/**
 * 검색(계좌번호같은사람정보보여주기)
 * @paramaccountNum		:비교할계좌번호(input값)
 * @return account 객체
 */
public Account get(String accountNum) {
```

```
		Account account = null;//검색된애를찾아주면그값을담는것

		for (int i = 0; i <count; i++) {
			if(accounts[i].getAccountNum().equals(accountNum)) {//  이건값을비교하기위한연산자
				account = accounts[i];
				break;
			}
		}
		return account;
}


/**
 * 이름으로검색하기
 *
 * @paramaccountOwner
 * @return
 */
public Account[] search(String accountOwner) {
		intcnt=0;//값이있는것만담아주기위한변수
		for (int i    = 0; i <count; i++) {
			if(accounts[i].getAccountOwner().equals(accountOwner)){//  이건값을비교하기위한연산
				cnt++;
			}
		}
		Account[] temp = new Account[cnt];
		cnt=0;//재사용, temp배열에값을넣기위해사용
		for (int i = 0; i <temp.length; i++) {
			if(accounts[i].getAccountOwner().equals(accountOwner)) {
				temp[cnt]=accounts[i];
				cnt++;
			}
		}
		return temp;
}

/**
 * 삭제
 *
 * @paramaccountNum
 * @return
 */
publicboolean remove(String accountNum) {
		booleanremoveTF= false;//삭제여부를알려주는변수
		for (int i    = 0; i <count; i++) {
```

```
                        if(accounts[i].getAccountNum().equals(accountNum)){
                                // 같은계좌가있으면뒤에값을앞으로담아주기
                                for (int j    = 0; j <count; j++) {
                                        accounts[j] = accounts[j+1];
                                }
                                accounts[count-1]=null;
                                //맨마지막값은  null값처리
                                count--;
                                //하나찾았으면찾은것은빼놓고다시찾기위해설정
                                removeTF = true;
                        }
                }

                return removeTF;//삭제가제대로됐으면  true, false
        }

}
```

```
/**
 * 은행계좌관리애플리케이션
```

```
 *
 * @author박소연
 *
 */
public class AMS {

	public static void main(String[] args) {
		//accountmanager생성하기
		AccountManager manager = new AccountManager(10);
		//AccountManager manager = new AccountManager(100);//이거생성자만들기

		//예시로받기, 기본입출금계좌
		Account account = new Account("1111-2222-3333", "김기정", 1111, 100000);
		manager.add(account);
		manager.add(new Account("2222-3333-4444", "박지성", 1111, 200000));
		manager.add(new Account("2222-3333-4444", "박지성", 1111, 200000));
		manager.add(new Account("2222-3333-4444", "박지성", 1111, 200000));
		manager.add(new Account("3333-4444-5555", "손흥민", 1111, 300000));

		//Up casting 대출계좌담기
		manager.add(new MinusAccount("9999-1111-2222", "김대출", 1111, 1000, 100000000));
		manager.add(new MinusAccount("9999-2222-2222", "홍길동", 1111, 1000, 100000000));

		//다시기본입출금계좌
		manager.add(new Account("4444-2222-3333", "박찬호", 1111, 200000));

		//잘등록되어있는지전체목록불러오기
		System.out.println("-------------------------계좌목록---------------------------");
		Account[] list =manager.list();
		for (Account account2 : list) {
			if(account2 instanceofMinusAccount) {
				//Account안됨!!
				//분기할때부모부터주면안된다!!
상속인경우에는자식클래스주소인데도부모랑같게취급하기때문
				System.out.println("마이너스계좌\t"+account2.toString());
			}else {
				System.out.println("입출금계좌\t"+account2.toString());
			}
		}
		System.out.println("-----------------------------------------------------------");
		System.out.println();

		//계좌검색한거보여주기
		System.out.println("-------------------------계좌검색---------------------------");
		Account search = manager.get("2222-3333-4444");
```

```
		if(search == null) {
			System.out.println("찾으시는계좌가없습니다.");
		}
		else {
			System.out.println("검색정보 :  "+search);
		}
		System.out.println("-------------------------------------------------------");
		System.out.println();

		//이름검색한거다보여주기
		System.out.println("-------------------이름검색-------------------");
		Account[] searchN = manager.search("박지성");

		for (Account account2 : searchN) {

			if(account2 != null) {

				System.out.println("계좌 :  "+account2.toString());
			}
		}
		System.out.println("-------------------------------------------------------");
		System.out.println();

		//계좌삭제하기
		System.out.println("-------------------계좌삭제-------------------");
		booleanremoveTF=manager.remove("000");
		if(removeTF) {
			System.out.println("계좌가삭제되었습니다.");
		}else {
			System.out.println("찾으시는계좌가존재하지않습니다.");
		}
		System.out.println("-------------------------------------------------------");



	}

}
```

```
/**
 * 동물을나타내는추상클래스
 *
 * @author박소연
 *
 */
public abstract class Animal {

	protected String name;
	protectedint age;

	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	publicintgetAge() {
		return age;
	}
	public void setAge(int age) {
		this.age = age;
	}

	//일반메소드
	public void printInfo() {
		System.out.println("이름 :"+ name+",  나이 : "+age);
	}

	//추상메소드
	public abstract void sleep();
	public abstract void eat();
}
```

```
/**
 * 자바표준 API의기본클래스사용하기
 */
public class APIExample {
	public static void main(String[] args) {
		//문자열만들기
		//명시적생성
		String str;
		str = new String("자바가싫어요...");//생성자호출

		//매번 new String을하기힘들어서이렇게한다
		//묵시적생성
		str ="자바가싫어요...";//이렇게해도자동으로 String인스턴스가만들어진다.

		int length= str.length();//length메소드사용, 문자열갯수세기
		System.out.println(length);

		//charAt사용하기

		char c = str.charAt(0);
		System.out.println(c);

		//Math클래스
		int x = -20;
		System.out.println(Math.abs(x));

	}
}
```

```
/**
 * 1차원배열선언, 생성, 초기화
 * @author박소연
 *
 */
public class ArrayExample {

	public static void main(String[] args) {

		/*array[0]=10;
		int[] array;
		array = new int[10];
		array[9]=100;


//		System.out.println(array[0]);
//		System.out.println(array[9]);
		System.out.println(array.length);

		for (int i = 0; i <array.length; i++) {
			System.out.print(array[i]+"\t");
		}
		//int[] array2 = new int[] {1,2,3,4,5};
		int[] array2= {1,2,3,4,5};
		for (int i    = 0; i < array2.length; i++) {
			System.out.println(array2[i]);
		}
```

```
//확장포문, 배열에만사용
for(int value : array2) {
		System.out.println(value);
}
for (int i : array2) {
		System.out.println(i);
}


 */

//미션1(배열복사하기)->이걸 7개로늘리기
int[] array3 = {3,1,9,2,5};
int[] array4 = new int[7];//만들어지고나서나머지두개비어있는상태로

//System.out.println("+++++++++++++++"+array3[1]);
for (int i = 0; i < array3.length; i++) {
/*
 * int a = array3[i];
 * array4[i] = a;
 * int b = array4[i];
 * System.out.println("i번째"+b);
*/
		array4[i] = array3[i];

}

for (int i : array4) {
		System.out.print(i+"\t");
}



//미션2(정렬)
int[] lottos = {34, 12, 3, 9, 25, 2};//출력했을때숫자작은숫자로정렬되게출력(오름차순), 새로만들지말고
//int start=0;
//  정렬코드
/*
for (int i = 5; i>0; i--) {
		for (int j = 0; j <5; j++) {
			if(lottos[j+1]<lottos[j]) {
				start = lottos[j+1];
				lottos[j+1] = lottos[j];
				lottos[j] =start;
			}
```

```
                        }

                }
                */
                for(int i =0; i<lottos.length; i++){
                        for(int j =0; j<lottos.length-1; j++){
                                int temp = 0;
                                if(lottos[j] >lottos[j+1]){
                                        temp = lottos[j];
                                        lottos[j]=lottos[j+1];
                                        lottos[j+1]=temp;
                                }
                        }
                  }
                for (int i : lottos) {
                        System.out.print(i +"\t");
                }

        }

}
```

```
/**
 * 다차원배열테스트
 * @author박소연
 *
 */
public class ArrayExample2 {


        public static void main(String[] args) {
                //1. 선언
                char[][] array;//여기에숫자넣으면절대안된다.
                //2. 생성
                array = new char[10][10];
                //3. 초기화
```

```
		array[0][0]='A';
		array[9][9]='Z';


		for (int i = 0; i <array.length; i++) {
			for (int j = 0; j < array[i].length; j++) {
				System.out.print(array[i][j]+"\t");
			}
			System.out.println();
		}

		int[][] array2 = new int[][] {{1,2,3},{4,5,6},{7,8,9}};
		//[3][3]짜리
		//new 뒤에사이즈넣으면안됨! 초기화동시에할때는값이들어가고있어서안됨!

		int[][] array3 = {{1,2,3},{4,5,6},{7,8,9}};//이렇게가능

		//구구단만들어보기
		int[][] gugudan = new int[8][9];
		for (int i = 0; i <gugudan.length; i++) {
			for (int j = 0; j <gugudan[i].length; j++) {
				gugudan[i][j]=(i+2)*(j+1);
				System.out.print((i+2)+"*"+(j+1)+"="+gugudan[i][j]+"\t");
			}
			System.out.println();
		}

	}

}
```

```
/**
 * 레퍼런스타입배열선언, 생성, 초기화
 *
 * @author박소연
 *
 */
```

```
public class ArrayExample3 {

	public static void main(String[] args) {

		//선언
		String[] teams;
		//생성
		teams = new String[10];
		//초기화
		teams[0]="두산베어스";//new String("두산베어스");
		teams[1]="sk와이번즈";
		teams[2]="한화이글즈";
		teams[9]="NC  다이노즈";

		for (int i = 0; i <teams.length; i++) {
			int count=0;
			if (teams[i] != null) {

				count = teams[i].length();
			}
			System.out.println((i + 1) + "위  : " + teams[i] + " (글자수  : " + count + ")");
		}

	}

}
```

```
importjava.util.Scanner;
```

```
/**
 * 레퍼런스타입배열선언, 생성, 초기화
 *
 * @author박소연
 *
 */
public class ArrayExample4 {

	public static void main(String[] args) {

		//Account account;
		//account = new Account("1111-2222-3333", "김기정", 1111, 100000);

		//배열선언및생성
		Account[] accounts = new Account [100];
		int index = 0;

		//은행계좌개설
		accounts[index] = new Account ("1111-2222-3333", "김기정", 1111, 100000);
		index++;
		accounts[index] = new Account ("2222-3333-4444", "박지성", 1111, 200000);
		index++;
		accounts[index] = new Account ("3333-4444-5555", "손흥민", 1111, 300000);
		index++;

		//전체계좌목록출력
		for (int i=0; i<index; i++) {
			System.out.println(accounts[i].toString());// 외부에서출력하는것을접근하지말고,
외부에서사용하는출력기능을만드는것이객체화
		}

		//계좌번호로계좌조회하는기능

		String num = null;
		Scanner sc = new Scanner(System.in);
		System.out.print("검색계좌번호 : ");
		num = sc.nextLine();

		Account account = null;//검색된애를찾아주면그값을담는것
		for (int i = 0; i <index; i++) {

			if(accounts[i].getAccountNum().equals(num)) {// 이건값을비교하기위한연산자
				/*
				 * Account acc = accounts[i]
```

```
                        * String numm = acc.getAccountNum();
                        * booleaneq = numm.equals(num);
                        * if(eq){
                        *                       account = acc;
                        *                       break;
                        * }
                        */
                        account = accounts[i];
                        break;
                }
        }
                if(account != null){
                        System.out.println(account.toString());
                }else {
                        System.out.println("찾으시는계좌가없습니다.");
                }

    }

}
```

```
/**
 * 레퍼런스타입배열선언, 생성, 초기화
 * 다차원배열
 *
 * @author박소연
 *
 */
public class ArrayExample5 {

	public static void main(String[] args) {

		String[][] array = new String[3][100];
		//3개는분류(야구팀, 축구팀, 농구팀), 100개는그안에들어가는문자열들(팀이름들)

		array[0][0] = "AAA";
		array[2][0] = "ZZZ";

		//한번에초기화하는방법
		String[][] array2 = new String[][] {{"a","b","c"},{"d","e","f"},{"g","h","i"}};
		//a, b, c 이게 100개나온다.

		String[][] array3 = {{"a","b","c"},{"d","e","f"},{"g","h","i"}};


	}
}
```

```
public class Bicycle/*extends Object*/ {

	int id;
	String brand;

	public Bicycle() {
		//super();  이렇게명시적으로안해도컴파일이자동으로호출
		this(0, null);
	}

	public Bicycle(int id, String brand){
		//super();  이렇게명시적으로안해도컴파일이자동으로호출
		this.id = id;
		this.brand = brand;
	}

	/**
	 * 달리는메소드
	 */
	public void running() {
		System.out.println("자전거가달립니다.");
	}

}
```

```
public class Calculator {

	//메소드오버로딩(중복정의)  예
	public static int sum(int x, int y) {
		returnx+y;
	}

	public static double sum(double x, double y) {
		returnx+y;
	}

	public static void main(String[] args) {
		//원래는이렇게호출해야한다.
		//Calculator.sum(20, 30);  여기는같은클래스라이렇게가능
		System.out.println(sum(20, 30));
		System.out.println(sum(20.5, 30.4));


	}
}
```

```
/**
 * Shape의속성과기능을상속받는자식클래스
 *
 * @author박소연
 *
 */
public class Circle extends Shape {

	private double radian;
	public Circle() {
		//super();
		this(0.0, 0.0, 0.0);
	}

	public Circle(double x, double y, double radian) {
		//super(x, y);
		//protected로해놨기때문에바로접근이가능하다
```

```
		this.x = x;
		this.y = y;
		this.radian = radian;
	}

	public double getRadian() {
		return radian;
	}
	public void setRadian(double radian) {
		this.radian = radian;
	}

	@Override
	public void draw() {
		//super.draw();//부모클래스의 draw를그대로사용할경우
		System.out.println(x+","+y+"," +getRadian()+"의원입니다.");
		//x랑  y가
	}

	@Override
	public double getLength() {
		return 2*Math.PI*radian;
	}

	@Override
	public double getArea() {
		returnMath.pow(radian, 2)*Math.PI;
	}

	@Override
	public String toString() {
		return "Circle [radian=" + radian + ", toString()=" + super.toString() + ", getClass()=" + getClass()
				+ ", hashCode()=" + hashCode() + "]";
	}

}
public class Dog extends Animal {

	privateboolean loyalty;

	public Dog() {
		this(null, 0, false);
	}

	public Dog(String name, int age, boolean loyalty) {
```

```
		this.name = name;
		this.age = age;
		this.loyalty = loyalty;
	}

	publicbooleanisLoyalty() {
		return loyalty;
	}

	public void setLoyalty(boolean loyalty) {
		this.loyalty = loyalty;
	}

	@Override
	public void sleep() {
		System.out.println("강아지가주무십니다...");

	}

	@Override
	public void eat() {
		System.out.println("강아지가먹습니다...");

	}

	public static void main(String[] args) {
		//클래스형변환(자동형변환  Up Casting)
		Animal animal = null;
		animal = new Dog("루니",3, false);//animal로참조
		animal.printInfo();//Animal에있는메소드
		animal.eat();
		animal.sleep();
	}

}
```

```
/**
 * java FileReader [읽고자하는파일명] [읽고자하는파일명]
```

```
 *
 * 이렇게실행했을때
 * @author박소연
 *
 */
public class FileReader {

	public static void main(String[] args) {
		if(args.length != 2) {
			System.out.println("사용법 : java FileReader [읽고자하는파일명] [읽고자하는파일명]");
			return;
		}
		String fileName1 = args[0];
		String fileName2 = args[1];

		System.out.println(fileName1);
		System.out.println(fileName2);
	}

}
```

```
public class InheritenceExample {

	public static void main(String[] args) {

		//부모클래스테스트
		Bicycle bicycle = new Bicycle(10, "삼천리");

		System.out.println(bicycle.brand);
		bicycle.running();

		//자식클래스테스트
		MountainBicyclemountainBicycle = new MountainBicycle(10, "삼천포", "카본", true);
		//재사용
		System.out.println(mountainBicycle.id);
		System.out.println(mountainBicycle.brand);
		//확장
		System.out.println(mountainBicycle.frame);
		System.out.println(mountainBicycle.suspension);

		//메소드오버라이딩테스트
		mountainBicycle.running();

	}
}
```

```
/**
 * Account를확장한예(상속)
 */
public class MinusAccount extends Account {

	//새로운속성 :대출금액
	private long borrowMoney;

	//기본생성자
	publicMinusAccount() {
		this(null, null, 0, 0, 0);
	}
	//생성자 :꼭부모까지만들기!
	publicMinusAccount(String accountNum, String accountOwner, intpasswd, long restMoney, long borrowMoney) {
		super(accountNum, accountOwner, passwd, restMoney);
		this.borrowMoney = borrowMoney;
	}

	//getter,setter
	public long getBorrowMoney() {
		returnborrowMoney;
	}
	public void setBorrowMoney(long borrowMoney) {
		this.borrowMoney = borrowMoney;
	}

	//메소드오버라이딩
	@Override
	public long getRestMoney() {
		returnsuper.getRestMoney()- getBorrowMoney();
		//부모클래스의restMoney가 private이라서 get으로가져오고,
		//지금자기getRestMoney가아닌부모의메소드라는뜻으로
		//super.이렇게붙여준다.
	}

	@Override
	public String toString() {
		returnsuper.toString()+"\t"+borrowMoney;
	}

	//테스트용어플리케이션클래스
	public static void main(String[] args) {
		MinusAccountminusAccount = new MinusAccount();
		System.out.println(minusAccount.getBorrowMoney());
```

```
		MinusAccount minusAccount2 = new MinusAccount("9999-1111-2222", "김대출", 1111, 10000, 10000000);
	//	MinusAccount2.deposit();
		System.out.println("현재잔액 : "+minusAccount2.getRestMoney());//재사용했더니, 대출금액은반영되지않았다.
		System.out.println(minusAccount2);//account의toString을접근해서가져옴,
여기에대출금액까지나왔으면좋겠어서여기에추가
	}
public class Mission {


	public static void main(String[] args) {
		Account account1 = new Account("1111-2222-3333", "김기정", 1111, 100000);
		Account account2 = new Account("1111-2222-3333", "김기정", 1111, 100000);

		//내용비교하기
		System.out.println(account1 == account2);//주소값이다르다.
		System.out.println();
		System.out.println(account1.equals(account2));//이게 true가떨어지게만들기. 같으려면어떻게해야하는가?



	}

}
```

```
public class MountainBicycle extends Bicycle{
	//Bicycle에있는 id랑 brand가존재하는상황
	//int id;
	//String brand;

	//새로추가된속성
	String frame; //재질
	boolean suspension;//완충작용해주는애

	publicMountainBicycle() {
		//super();
		this(null, false);
	}
	publicMountainBicycle(String frame, boolean suspension) {
		//super();
		this.frame = frame;
		this.suspension = suspension;
	}

	publicMountainBicycle(int id, String brand, String frame, boolean suspension) {
		//this.brand=brand;
		//this.id = id;
		//코드중복, 이건이렇게
		super(id, brand);
		//여기는 super(); 이거가없음!!!

		this.frame = frame;
		this.suspension = suspension;
	}

	/**
	 * Bicycle에있는 running 메소드재정의(메소드오버라이딩)
	 * 오버로딩과헷갈림(중복정의, 매개변수값타입, 갯수가다른경우)
	 *
	 */
	public void running() {
		System.out.println("산도달립니다.");
	}



}
```

```
/**
 * Shape를활용한다형성예제
 *
 * @author박소연
 *
 */
public class PolymophismExample {

	public static void main(String[] args) {
		//  클래스형변환(Up casting)
		Shape shape = null;
		//shape = new Shape();//이렇게하는데보통
		shape = new Circle(10, 10, 20);//같은변수에다른객체들을참조해서사용할수있음
		System.out.println(shape);

		shape = new Rectangle(10, 10, 50,20);
		System.out.println(shape);

		//궁극적으로  shape는  Shape클래스를가르킨다.
		System.out.println(shape.x); //자기꺼라서가져옴. 10이렇게찍힘
		//System.out.println(shape.getWidth());//에러가난다.  밑에숨어있어서

		System.out.println(shape.getArea());//오버라이딩된거는또된다,  제대로계산되서  1000나옴


		//추가된속성이나메소드를접근하기위해  Down Casting  필요
		Rectangle rectangle = (Rectangle)shape;
		System.out.println(rectangle.getWidth());


		System.out.println(shape instanceof Object);
		System.out.println(shape instanceof Shape);
		System.out.println(shape instanceof Rectangle);
		//System.out.println(shape instanceof String);//shape에안담겨있어서오류


	}

}
```

```
public class Rectangle extends Shape {

	private double width, height;

	public Rectangle() {
		//super();
		this(0.0, 0.0, 0.0, 0.0);
	}

	public Rectangle(double x, double y, double width, double height) {
		//super(x, y);
		this.x = x;
		this.y = y;
		this.width = width;
		this.height = height;
	}

	public double getWidth() {
		return width;
	}

	public void setWidth(double width) {
		this.width = width;
	}

	public double getHeight() {
		return height;
	}

	public void setHeight(double height) {
		this.height = height;
	}

	@Override
	public void draw() {
		System.out.println(x+","+y+","+getWidth()+","+getHeight()+"의사각형입니다.");
	}

	@Override
	public double getLength() {
		return (getWidth()+getHeight())*2;
	}

	@Override
	public double getArea() {
```

```
                returngetWidth()*getHeight();
        }

        @Override
        public String toString() {
                return "Rectangle [width=" + width + ", height=" + height + ", x=" + x +", y=" + y +", toString()=" + super.toString() + ",
getClass()="
                                + getClass() + ", hashCode()=" + hashCode() + "]";
        }
/**
 *
 * @author박소연
 *
 * 모든도형의공통적인속성과기능을정의한추상클래스
 */
public abstract class Shape {

        //인스턴스가생성이안되는데굳이은닉화할필요없음
        //생략하면얘를상속받는애들이같은패키지에서만받는다
        //그래서protected :패키지가같거나다르더라도상속관계면사용가능
        protected double x, y;

        //구현메소드를추상메소드로선언하기, Shape는이제 new 사용해서생성안됨
        //서브클래스가반드시구현(재정의)해야할수직적규약
        public abstract void draw();

        public abstract double getLength();

        public abstract double getArea();

        //public void xxx(){}
        //이런식으로추상메소드가아닌일반적인메소드가있어도됨

        @Override
        public String toString() {
                return x+","+y+"의도형입니다...";
        }
}
```

```
public class ShapeApp {

	public static void main(String[] args) {
		//부모클래스테스트
		//추상클래스가되면서생성은불가능
		//Shape shape = new Shape(12.5, 35.7);//의존관계는이렇게메소드에서클래스를호출한다
		Shape shape = null;//선언과사용만가능하다.
		shape.draw();

		//자식클래스테스트
		Circle circle = new Circle(15.0,15.0,30);
		circle.draw();
		System.out.println(circle.getLength());
		System.out.println(circle.getArea());

		Rectangle rectangle = new Rectangle(15.0, 15.0, 3, 4);
		rectangle.draw();
		System.out.println(rectangle.getLength());
		System.out.println(rectangle.getArea());

		System.out.println(shape);
		//toString을오버라이딩해놓으면레퍼런스변수의주소값을알려주는데,
		//이걸이용해서바로출력이가능하게만들수있음
		System.out.println(shape.toString());

		String str = "Java Programming";
		System.out.println(str); //주소값이아닌 java Programming을출력한다
		//원래는 Object에있는toString()을해서주소값이나와야하는데
		//String클래스가toString()을오버라이딩한다.
		//그래서실제값이나오는거

		System.out.println(circle);
	}


/**
 * LIFO 구조의스택구현
 *
 * @author박소연
 *
 */
public class Stack {

	privateint[] array;
	private int count;//어디에쌓이고있는지위치값
```

```
//기본생성자
public Stack() {
		//만약에값을전달안했을때는일단만들어는줘야하기때문에
		//이렇게매개변수를받는생성자로넘겨줘야한다.
		this(50);
}
//int타입의매개변수를받는생성자
public Stack(int size) {
		this.array = new int[size];
}

//getter함수와  setter함수
publicintgetCount() {
		return count;
}
public void setCount(int count) {
		this.count = count;
}
publicint[] getArray() {
		return array;
}
public void setArray(int[] array) {
		this.array = array;
}

/**
 * 스택에값을저장하는메소드
 *
 * @paramvalue		:저장할값
 */
public void push(int value) {
		array[count] = value;//들어오는데이터를저장
		count++;
}

/**
 * 스택저장값을추출하는메소드
 *
 * @return		:		추출된값
 */
publicint pop() {
		/*
		 * int result = array[count-1];//중복코드가발생함
```

```
		array[count-1]=0;//0으로할당안해도상관없음
		count--;
		return result;
		*/

		//중복코드제거한후더간단한코드
		int result = array[--count];
		array[count] = 0;
		return result;
	}

	/**
	 * 스택의갯수세는메소드
	 *
	 * @return :스택에저장된값갯수
	 */
	publicint length() {
		return count;
	}

	//테스트
	public static void main(String[] args) {
		//Stack stack = new Stack(100);//생성자호출 - 예시)스택을 100개줄거다.
		Stack stack = new Stack(10);

		//push메소드테스트
		stack.push(5);//stack에 5를담는다.
		stack.push(1);
		stack.push(9);

		//pop메소드테스트
		System.out.println("추출된값 : "+stack.pop());
		System.out.println("추출된값 : "+stack.pop());
		System.out.println("추출된값 : "+stack.pop());

		//length메소드테스트
		System.out.println("길이 : "+stack.length());
	}

}
```

```
public class Sword implements Weapon {

	@Override
	public void attack() {

		System.out.println("장칼로공격합니다...");
	}

}
public class Unit {
	//연관, Has a  관계이기때문에꼭선언을해줘야한다.
	private Weapon weapon;//필요에따라서받아서지속적으로사용한다.
	//생성이될때생성자에서초기화할필요가없다.

	//만약에집합관계로되어있으면
	/*
	 * public Unit(Weapon weapon, String name){
	 *			this.weapon = weapon;
	 *			this.name = name;
	 * }
	 */

	//만약에복합관계로되어있으면
	/*
	 * public Unit(String name){
	 *			weapon = new weapon();
	 *			this.name = name;
	 * }
	 */

	private String name;//생성자만들때이것만초기화할것
	//전체와부분의관계가아니라서그냥  getter setter만만들면된다
	public Unit(String name) {
		this.name = name;
	}
	public Weapon getWeapon() {
		return weapon;
	}
	public void setWeapon(Weapon weapon) {
		this.weapon = weapon;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
```

```
		this.name = name;
	}

	//여기 attack은 Weapon에있는거랑다르다
	public void attack() {
		//인터페이스가없다면 if(weapon ==)이런식으로써야함, 규격이있어서이렇게가능
		weapon.attack();//추상메소드사용한다
	}

	//테스트용
	public static void main(String[] args) {


		Unit unit = new Unit("SCV");
		//인터페이스도추상클래스처럼이렇게선언할수있다.
		Weapon weapon = null;
		weapon = new Gun();//업케스팅

		unit.setWeapon(weapon);
		unit.attack();

		weapon = new Sword();
		unit.setWeapon(weapon);
		unit.attack();

		/*
		 * Weapon weapon = null;
		 * weapon = new Gun();
		 * weapon.attack();
		 *  이렇게해도가능함
		 */

	}

}


/**
 *  무기에대한수평적규격역할의인터페이스
 *
 * @author박소연
 */
public interface Weapon {
	int WEIGHT = 10;//자동으로상수처리가된다.
	//public static final wieght = 10;
```

```
	public void attack();

}


public class Gun implements Weapon{

	@Override
	public void attack() {
		System.out.println("총으로공격합니다...");
	}
}
```