[06. 클래스]

1. 객체와 클래스에 대한 설명으로 틀린 것은 무엇입니까? 3번

➊ 클래스는 객체를 생성하기 위한 설계도(청사진)와 같은 것이다.

➋ new 연산자로 클래스의 생성자를 호출함으로써 객체가 생성된다.

➌ 하나의 클래스로 하나의 객체만 생성할 수 있다.

➍ 객체는 클래스의 인스턴스이다.

2. 클래스의 구성 멤버가 아닌 것은 무엇입니까? 4번

➊ 필드(field )

➋ 생성자(constructor )

➌ 메소드(method)

➍ 로컬 변수(local variable )

3. 필드, 생성자, 메소드에 대한 설명으로 틀린 것은 무엇입니까? 4번

➊ 필드는 객체의 데이터를 저장한다.

➋ 생성자는 객체의 초기화를 담당한다.

➌ 메소드는 객체의 동작 부분으로, 실행 코드를 가지고 있는 블록이다.

➍ 클래스는 반드시 필드와 메소드를 가져야 한다.

4. 필드에 대한 설명으로 틀린 것은 무엇입니까? 3번

➊ 필드는 메소드에서 사용할 수 있다.

➋ 인스턴스 필드 초기화는 생성자에서 할 수 있다.

➌ 필드는 반드시 생성자 선언 전에 선언되어야 한다.

➍ 필드는 초기값을 주지 않더라도 기본값으로 자동 초기화된다.

5. 생성자에 대한 설명으로 틀린 것은 무엇입니까? 1번

➊ 객체를 생성하려면 생성자 호출이 반드시 필요한 것은 아니다.

➋ 생성자는 다른 생성자를 호출하기 위해 this ( )를 사용할 수 있다.

➌ 생성자가 선언되지 않으면 컴파일러가 기본 생성자를 추가한다.

➍ 외부에서 객체를 생성할 수 없도록 생성자에 private 접근 제한자를 붙일 수 있다.

6. 메소드에 대한 설명으로 틀린 것은 무엇입니까? 4번

➊ 리턴값이 없는 메소드는 리턴 타입을 void로 해야 한다.

➋ 리턴 타입이 있는 메소드는 리턴값을 지정하기 위해 반드시return 문이 있어야 한다.

➌ 매개값의 수를 모를 경우 “…”를 이용해서 매개변수를 선언할 수 있다.

➍ 메소드의 이름은 중복해서 선언할 수 없다.

7. 메소드 오버로딩에 대한 설명으로 틀린 것은 무엇입니까? 2번

➊ 동일한 이름의 메소드를 여러 개 선언하는 것을 말한다.

➋ 반드시 리턴 타입이 달라야 한다.

➌ 매개변수의 타입, 수, 순서를 다르게 선언해야 한다.

➍ 매개값의 타입 및 수에 따라 호출될 메소드가 선택된다.

8. 인스턴스 멤버와 정적 멤버에 대한 설명으로 틀린 것은 무엇입니까? 2번

➊ 정적 멤버는 static으로 선언된 필드와 메소드를 말한다.

➋ 인스턴스 필드는 생성자 및 정적 블록에서 초기화될 수 있다.

➌ 정적 필드와 정적 메소드는 객체 생성 없이 클래스를 통해 접근할 수 있다.

➍ 인스턴스 필드와 메소드는 객체를 생성하고 사용해야 한다.

9. final 필드와 상수(static final )에 대한 설명으로 틀린 것은 무엇입니까? 2번

➊ final 필드와 상수는 초기값이 저장되면 값을 변경할 수 없다.

➋ final 필드와 상수는 생성자에서 초기화될 수 있다.

➌ 상수의 이름은 대문자로 작성하는 것이 관례이다.

➍ 상수는 객체 생성 없이 클래스를 통해 사용할 수 있다.

10. 패키지에 대한 설명으로 틀린 것은 무엇입니까? 4번

➊ 패키지는 클래스들을 그룹화시키는 기능을 한다.

➋ 클래스가 패키지에 소속되려면 패키지 선언을 반드시 해야 한다.

➌ import 문은 다른 패키지의 클래스를 사용할 때 필요하다.

➍ com.mycom 패키지에 소속된 클래스는 com.yourcom에 옮겨 놓아도 동작한다.

11. 접근 제한에 대한 설명으로 틀린 것은 무엇입니까? 3번

➊ 접근 제한자는 클래스, 필드, 생성자, 메소드의 사용을 제한한다.

➋ public 접근 제한은 아무런 제한 없이 해당 요소를 사용할 수 있게 한다.

➌ default 접근 제한은 해당 클래스 내부에서만 사용을 허가한다.

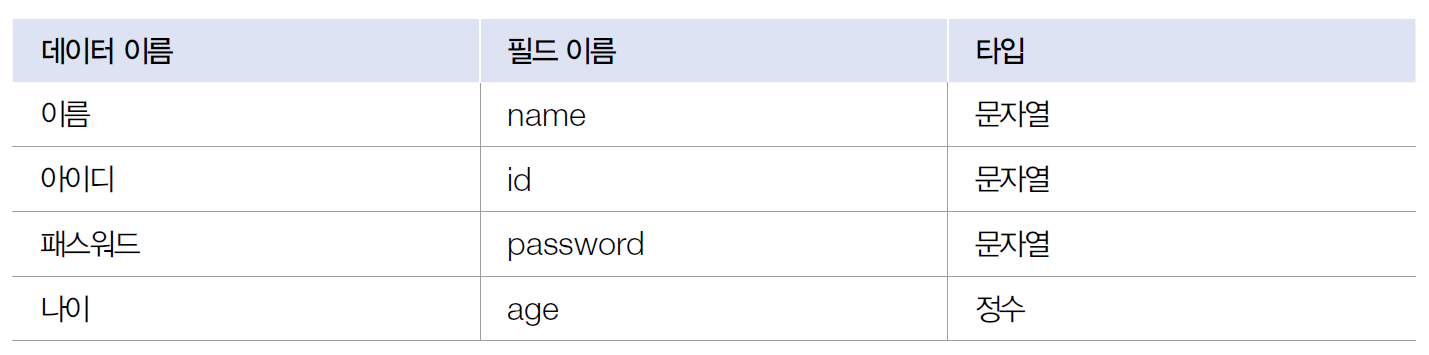
➍ 외부에서 접근하지 못하도록 하려면 private 접근 제한을 해야 한다.

12. 다음 클래스에서 해당 멤버가 필드, 생성자, 메소드 중 어떤 것인지( ) 안에 적어보세요.

| public class Member {  private String name; —---> ( 필드 )  public Member(String name) { … } —---> ( 생성자 )  public void setName(String name) { … } —---> ( 메소드 )  } |
| --- |

13. 현실 세계의 회원을 Member 클래스로 모델링하려고 합니다. 회원의 데이터로는 이름, 아이디,

패스워드, 나이가 있습니다. 이 데이터들을 가지는 Member 클래스를 선언해보세요.



14. 13번 문제에서 작성한 Member 클래스에 생성자를 추가하려고 합니다. 다음과 같이 name

필드와 id 필드를 외부에서 받은 값으로 초기화하도록 생성자를 선언해보세요.

| Member user1 = new Member("홍길동", "hong"); |
| --- |

public class Member {

String name;

String id;

String password;

int age;

}

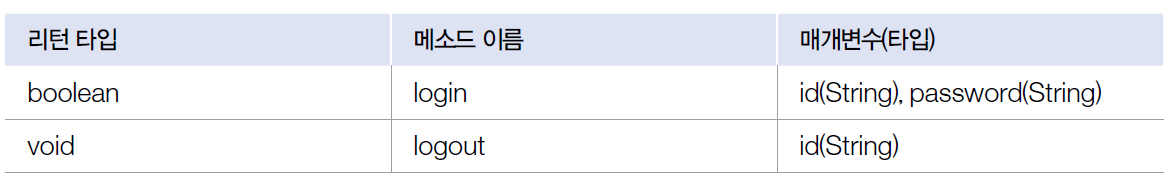
15. login ( ) 메소드를 호출할 때에는 매개값으로 id와 password를 제공하고, logout ( ) 메소드는

id만 매개값으로 제공하려고 합니다. 다음 조건과 예제 코드를 보고 MemberService 클래스에서

login ( ), logout ( ) 메소드를 선언해보세요.

➊ login ( ) 메소드는 매개값 id가 "hong", 매개값 password가 "12345" 일 경우에만 true로 리턴

➋ logout ( ) 메소드는 id + "님이 로그아웃 되었습니다"가 출력되도록 할 것



| public class Member {  String name;  String id;  String password;  int age;  Member(String name, String id) {  this.name = name;  this.id = id;  }  }  public class MemberService {  boolean login(String id, String password) {  if (id.equals("hong") && password.equals("12345")) {  return true;  } else {  return false;  }  }  void logout(String id) {  System.out.println("로그아웃 되었습니다.");  }  } |
| --- |

16. 다음은 컴퓨터 게임의 병사(marine)를 클래스로 정의한 것이다. 이 클래스의 멤버 중에 static을 붙여야

하는 것은 어떤 것들이고 그 이유는 무엇인가? (단, 모든 병사의 공격력과 방어력은 같아야 한다.)

| class Marine {  int x=0, y=0; // Marine의 위치좌표(x,y)  int hp = 60; // 현재 체력  static int weapon = 6; // 공격력  static int armor = 0; // 방어력  static void weaponUp() {  weapon++;  }  static void armorUp() {  armor++;  }  void move(int x, int y) {  this.x = x; // this.x는 인스턴스 변수, x는 지역변수  this.y = y; // this.y는 인스턴스 변수, y는 지역변수  }  } |
| --- |

\*풀이

weapon, armor - 모든 Marine인스턴스에 대해 동일한 값이어야 하므로.

weaponUp(), armorUp() - static변수에 대한 작업을 하는 메서드이므로.

17. 다음 중 아래의 add메서드를 올바르게 오버로딩 한 것은? (모두 고르시오) 답:2,3,4

| long add(int a, int b) { return a+b;} |
| --- |

➊ long add(int x, int y) { return x+y;}

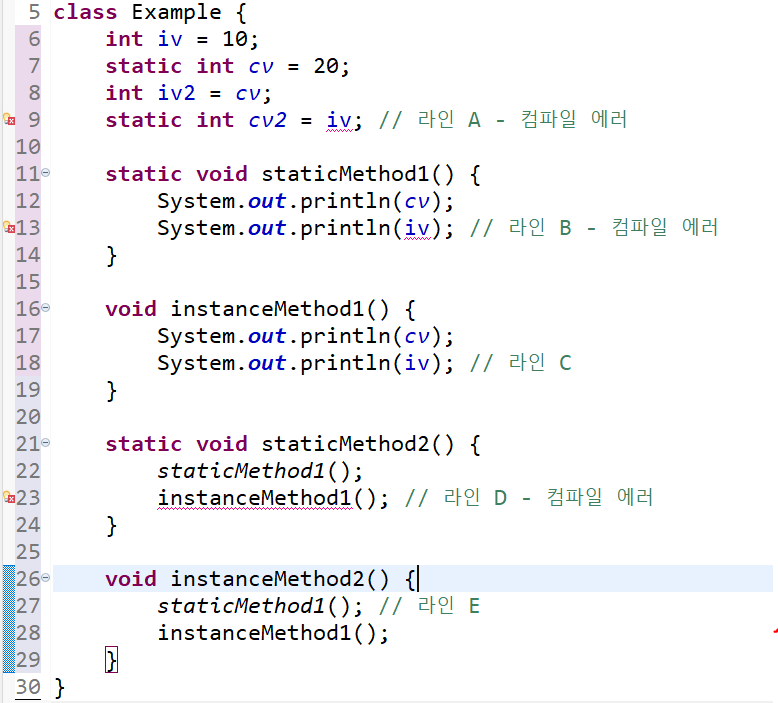
➋ long add(long a, long b) { return a+b;}

➌ int add(byte a, byte b) { return a+b;}

➍ int add(long a, int b) { return (int)(a+b);}

\*풀이: 1번은 매개변수의 타입이 같다.

18. 다음 코드를 실행하면 컴파일 에러가 발생한다. 에러가 발생하는 이유를 설명하시오.



정답:

첫번째 - static변수의 초기화에 인스턴스변수를 사용할 수 없다.

꼭 사용해야한다면, 객체를 생성해야한다.

두번째 - static메서드에서는 인스턴스변수를 사용할 수 없다.

세번째 - static메서드에서는 인스턴스메서드를 사용할 수 없다.

19. Tv클래스를 주어진 로직대로 완성하시오. 완성한 후에 실행해서 주어진 실행결과와 일치하는지

확인하라.

// (1) isPowerOn의 값이 true면 false로, false면 true로 바꾼다.

isPowerOn = !isPowerOn;

// (2) volume의 값이 MAX\_VOLUME보다 작을 때만 값을 1증가시킨다.

if (volume < MAX\_VOLUME)

volume++;

// (3) volume의 값이 MIN\_VOLUME보다 클 때만 값을 1감소시킨다.

if (volume > MIN\_VOLUME)

volume--;

// (4) channel의 값을 1증가시킨다.

// 만일 channel이 MAX\_CHANNEL이면, channel의 값을 MIN\_CHANNEL로 바꾼다.

if (channel == MAX\_CHANNEL) {

channel = MIN\_CHANNEL;

} else {

channel++;

}

// (5) channel의 값을 1감소시킨다.

// 만일 channel이 MIN\_CHANNEL이면, channel의 값을 MAX\_CHANNEL로 바꾼다.

if (channel == MIN\_CHANNEL) {

channel = MAX\_CHANNEL;

} else {

channel--;

}

20. 다음과 같이 정의된 메서드를 작성하고 테스트하시오.

-메서드명 : isNumber

-기능 : 주어진 문자열이 모두 숫자로만 이루어져있는지 확인한다.

모두 숫자로만 이루어져 있으면 true를 반환하고, 그렇지 않으면 false를 반환한다.

만일 주어진 문자열이 null이거나 빈문자열“”이라면 false를 반환한다.

-반환타입 : boolean

-매개변수 : String str ->검사할 문자열

| class Example {  public static boolean isNumber(String str) {  if (str == null || str.equals(""))  return false;  for (int i = 0; i < str.length(); i++) {  char ch = str.charAt(i);  if (ch < '0' || ch > '9') {  return false;  }  } // for  return true;  }  public static void main(String[] args) {  String str = "123";  System.out.println(str + "는 숫자입니까? " + isNumber(str));  str = "1234o";  System.out.println(str + "는 숫자입니까? " + isNumber(str));  }  } |
| --- |

21. 다음과 같이 정의된 메소드를 작성하고 테스트 하시오.

-메서드명 : max

-기능 : 주어진 int형 배열의 값 중에서 제일 큰 값을 반환한다. 만일 주어진 배열이 null이거나 크기가 0인 경우, -999999를 반환한다.

-반환타입 : int

-매개변수 : int[] arr -> 최대값을 구할 배열

| class Example {  public static int max(int[] arr) {  if (arr == null || arr.length == 0)  return -999999;  int max = arr[0]; // 배열의 첫 번째 값으로 최대값을 초기화 한다.  for (int i = 1; i < arr.length; i++) { // 배열의 두 번째 값부터 비교한다.  if (arr[i] > max)  max = arr[i];  }  return max;  }  public static void main(String[] args) {  int[] data = { 3, 2, 9, 4, 7 };  System.out.println(java.util.Arrays.toString(data));  System.out.println("최대값:" + max(data));  System.out.println("최대값:" + max(null));  System.out.println("최대값:" + max(new int[] {})); // 크기가 0인 배열  }  } |
| --- |

22. 은행 계좌 객체인 Account 객체는 잔고(balance) 필드를 가지고 있습니다. balance 필드는

음수값이 될 수 없고, 최대 백만 원까지만 저장할 수 있습니다. 외부에서 balance 필드를 마음대로

변경하지 못하도록 하고, 0 <= balance <= 1,000,000 범위의 값만 가질 수 있도록 Account 클래

스를 작성해보세요.

➊ Setter와 Getter를 이용

➋ 0과 1,000,000은 MIN\_BALANCE와 MAX\_BALANCE 상수를 선언해서 이용

➌ Setter의 매개값이 음수이거나 백만 원을 초과하면 현재 balance 값을 유지

| public class Account {  public static final int MIN\_BALANCE = 0;  public static final int MAX\_BALANCE = 1000000;  private int balance;  public int getBalance() {  return balance;  }  public void setBalance(int balance) {  if (balance < Account.MIN\_BALANCE || balance > Account.MAX\_BALANCE) {  return;  }  this.balance = balance;  }  } |
| --- |

23. 다음은 키보드로부터 계좌 정보를 입력받아 계좌를 관리하는 프로그램입니다. 계좌는A ccount

객체로 생성되고 BankApplication에서 길이 100인 Account[ ] 배열로 관리됩니다. 실행 결과를

보고, Account와 BankApplication 클래스를 작성해보세요(키보드로 입력받을 때는 Scanner

의 nextLine ( ) 메소드를 사용).

[Account.java]

| public class Account {  private String ano;  private String owner;  private int balance;  public Account(String ano, String owner, int balance) {  this.ano = ano;  this.owner = owner;  this.balance = balance;  }  public String getAno() {  return ano;  }  public void setAno(String ano) {  this.ano = ano;  }  public String getOwner() {  return owner;  }  public void setOwner(String owner) {  this.owner = owner;  }  public int getBalance() {  return balance;  }  public void setBalance(int balance) {  this.balance = balance;  }  } |
| --- |

[BankApplication.java]

| import java.util.Scanner;  public class BankApplication {  private static Account[] accountArray = new Account[100];  private static Scanner scanner = new Scanner(System.in);  public static void main(String[] args) {  boolean run = true;  while (run) {  System.out.println("----------------------------------------------------------");  System.out.println("1.계좌생성 | 2.계좌목록 | 3.예금 | 4.출금 | 5.종료");  System.out.println("----------------------------------------------------------");  System.out.print("선택> ");  int selectNo = scanner.nextInt();  if (selectNo == 1) {  createAccount();  } else if (selectNo == 2) {  accountList();  } else if (selectNo == 3) {  deposit();  } else if (selectNo == 4) {  withdraw();  } else if (selectNo == 5) {  run = false;  }  }  System.out.println("프로그램 종료");  }  // 계좌생성하기  private static void createAccount() {  System.out.println("--------------");  System.out.println("계좌생성");  System.out.println("--------------");  System.out.print("계좌번호: ");  String ano = scanner.next();  System.out.print("계좌주: ");  String owner = scanner.next();  System.out.print("초기입금액: ");  int balance = scanner.nextInt();    Account newAccount = new Account(ano, owner, balance);    for(int i=0; i<accountArray.length; i++) {  if(accountArray[i] == null) {  accountArray[i] = newAccount;  System.out.println("결과: 계좌가 생성되었습니다.");  break;  }  }  }  // 계좌목록보기  private static void accountList() {  System.out.println("--------------");  System.out.println("계좌목록");  System.out.println("--------------");  for (int i = 0; i < accountArray.length; i++) {  Account account = accountArray[i];  if (account != null) {  System.out.print(account.getAno());  System.out.print(" ");  System.out.print(account.getOwner());  System.out.print(" ");  System.out.print(account.getBalance());  System.out.println();  }  }  }  // 예금하기  private static void deposit() {  System.out.println("--------------");  System.out.println("예금");  System.out.println("--------------");  System.out.print("계좌번호: ");  String ano = scanner.next();  System.out.print("예금액: ");  int money = scanner.nextInt();  Account account = findAccount(ano);  if (account == null) {  System.out.println("결과: 계좌가 없습니다.");  return;  }  account.setBalance(account.getBalance() + money);  System.out.println("결과: 예금이 성공되었습니다.");  }  // 출금하기  private static void withdraw() {  System.out.println("--------------");  System.out.println("출금");  System.out.println("--------------");  System.out.print("계좌번호: ");  String ano = scanner.next();  System.out.print("출금액: ");  int money = scanner.nextInt();  Account account = findAccount(ano);  if (account == null) {  System.out.println("결과: 계좌가 없습니다.");  return;  }  account.setBalance(account.getBalance() - money);  System.out.println("결과: 출금이 성공되었습니다.");  }  // Account 배열에서 ano와 동일한 Account 객체 찾기  private static Account findAccount(String ano) {  Account account = null;  for (int i = 0; i < accountArray.length; i++) {  if (accountArray[i] != null) {  String dbAno = accountArray[i].getAno();  if (dbAno.equals(ano)) {  account = accountArray[i];  break;  }  }  }  return account;  }  } |
| --- |