Java Programming 이론

1. 자바의 기본(primitive) 데이터형의 종류를 나열하고 간단하게 설명하시오.

자바의 기본 데이터형은 boolean, byte, short, int, long, float, double, char가 있습니다.
boolean: 논리적인 값으로 true 또는 false를 나타냅니다.
byte: 8비트 정수 값을 나타냅니다.
short: 16비트 정수 값을 나타냅니다.
int: 32비트 정수 값을 나타냅니다.
long: 64비트 정수 값을 나타냅니다.
float: 32비트 부동소수점 숫자 값을 나타냅니다.
double: 64비트 부동소수점 숫자 값을 나타냅니다.
char: 16비트 유니코드 문자를 나타냅니다.

2. 다음의 코드를 컴파일할 때, 오류가 나는 이유를 적으시오

```
public class Test{
        public static void main(String[] args){
                int x;
                System.out.println(x);
        }
}
```

변수 x를 초기화하지 않고 사용하였기에 오류가 발생하였습니다.
x는 지역 변수는 초기화되지 않은 상태에서 사용할 수 없습니다. 초기화 하지 않은 상태에서 컴파일러를 하여 오류를 감지하고 해당 라인에서 컴파일 오류를 발생시킵니다.
int x = 0;과 같이 초기값을 지정해주면 오류가 해결됩니다.

3. 자바에서 다음 클래스를 컴파일 한 후 실행 결과는?

```
public class Test {
    public static void main(String[] args) {
            String city = "서울";
            String country = "";
            switch(city) {
            case "서울" :
            case "부산" : country = "한국";
            case "북경" : country = "중국";
             case "동경" : country = "일본";     break;
              default : country = "대상없음";
           }
                System.out.println(country);
    }
}
```

컴파일 후 실행 결과는 "일본"이 출력됩니다.

switch~case 문에서 break가 있는 case값인 “일본”이 출력이 됩니다.
default는 해당되는 case가 없을 때 출력이 되는 구문입니다.

4. 접근 제어자(Access Modifier)는 클래스와 클래스의 멤버 접근 대한 접근 정도를 지정할 때 사용한다. 클래스 멤버의 접근지정자를 4가지를 나열하라.

접근 제어자는 클래스, 멤버 변수, 메서드, 생성자 등의 가시성을 조절하는 데 사용됩니다.
접근 제어자는 public, protected, private, default 4가지가 있습니다.
public: 해당 멤버는 어디서든 접근할 수 있습니다. 다른 클래스에서도 접근이 가능합니다.
protected: 해당 멤버는 동일한 패키지 내의 클래스와 상속 관계에 있는 클래스에서 접근할 수 있습니다.
private: 해당 멤버는 동일한 클래스 내에서만 접근할 수 있습니다. 다른 클래스에서 접근이 불가능합니다.
default (접근 지정자를 명시하지 않음): 해당 멤버는 동일한 패키지 내의 클래스에서만 접근할 수 있습니다. 다른 패키지의 클래스에서는 접근이 불가능합니다.
이러한 접근 지정자를 사용하여 클래스 멤버의 가시성과 접근 범위를 제어할 수 있습니다.

5. 다음은 자바 예약어(keyword)에 대한 설명이다. (ㄱ)와 (ㄴ)에 들어갈 적절한 예약어를 쓰시오.

> 클래스 접근 지정자(Modfier)로 (ㄱ) 명시되면 , 다른 클래스들은 해당 클래스를 상속 받을 수 없다. 메서드의 Modifier로 (ㄱ) 지정되면 자식 클래스는 그 메서드를 오버라이딩 할 수 없다.
> (ㄴ) 메소드가 하나 이상 존재하는 클래스는 클래스 선언부에 (ㄴ) 제어자(Modifier)가 선언되어야 한다. (ㄴ) 메소드가 존재하는 클래스를 상속받는 자식 클래스에서는 해당 (ㄴ)메서드를 구현하거나 해당 클래스의 선언부에 (ㄴ) 제어자(Modifier)가 명시되어야 한다.

(ㄱ) : final
(ㄴ) : abstract

6. 다음이 설명하는 특징을 가진 컬렉션 인터페이스는?

> ㄱ. Key와 Value의 매핑으로 관리된다.
> ㄴ. Key는 중복될 수 없다.
> ㄷ. Collection 인터페이스를 상속하지는 않는다.

Map 인터페이스

Map은 Key와 Value의 쌍으로 데이터를 저장하고 관리합니다.
각 Key는 하나의 Value와 매핑되어 있습니다.
Key는 중복될 수 없습니다. Map은 각 Key는 고유해야 하며, 중복된 Key를 가질 수 없습니다.
동일한 Key를 사용하면 기존의 Value가 대체됩니다.
Collection 인터페이스를 상속하지 않습니다.
Map 인터페이스는 Collection 인터페이스를 직접 상속하지 않지만, Java의 컬렉션 프레임워크에 속합니다.

7. 자바에서 아래의 char형 배열 alpha를 모두 출력하고자 한다. 빈칸에 적절한 코드를 삽입하라.

(1)

```
char[] alpha = { 'a', 'b', 'c', 'd' };
for (                                    ) {
    System.out.print(alpha[i]);
}
```

(2)

```
char[] alpha = { 'a', 'b', 'c', 'd' };
for (                                ) {
    System.out.print(c);
}
```

(1) (int i = 0; i < alpha.length; i++)
for를 이용한 반복문으로 출력합니다.

(2) (char c : alpha)
for~each문을 통해서 alpha의 배열을 각각 출력합니다.

8. 다음 코드의 실행 결과 나타 나도록 빈곳에 적당한 코드를 작성하시오.

```
class Person{
    String name;
    int age;

    public Person(String name, int age){
      this.name=name;
      this.age=age;
    }

    public void show(){
      System.out.println(name+" "+age+" ");
    }
}

class Student extends Person{
   String major;
   (ㄱ) //  생성자 작성
   (ㄴ) // show() 함수 오버라이딩

}

public class Test {
     public static void main(String[] args) {
           Person p=new Student("홍길동", 20, "컴퓨터공학");
           p.show();
      }
}
```

출력결과 : 홍길동 20 컴퓨터공학

```
package com.test;

class Person{
    String name;
    int age;
    String major;

    public Person(String name, int age){
      this.name=name;
      this.age=age;
    }

    public void show(){
      System.out.println(name+" "+age+" ");
      System.out.println(major);
    }
}

class Student extends Person{

(ㄱ) public Student(String name, int age, String major) {
                super(name, age);
                this.major=major;

        }
  (ㄴ) @Override
        public void show() {
                // TODO Auto-generated method stub
                super.show();
        }
}

public class Test {
     public static void main(String[] args) {
            Person p=new Student("홍길동", 20, "컴퓨터공학");
            p.show();
      }
}
```

9. 다형성에 대한 설명 중 틀린 것은?

① 추상 메소드를 두는 이유는 상속 받는 클래스에서 다형성을 실현하도록 하기 위함이다.
② 인터페이스도 구현하는 클래스에서 다형성을 실현하도록 하기 위함이다.
③ 다형성은 서브클래스들이 수퍼 클래스의 동일한 메소드를 서로 다르게 오버라이딩하여 이루어진다.
④ 자바에서 다형성은 모호한(ambiguous) 문제를 일으키므로 사용하지 않는 것이 바람직하다.

④ 자바에서 다형성은 모호한(ambiguous) 문제를 일으키므로 사용하지 않는 것이 바람직하다.
다형성은 오히려 코드의 유연성과 확장성을 높이는데 기여합니다. 다형성을 이용하면 다른 클래스의 객체를 동일한 타입으로 다룰 수 있으며, 이를 통해 인터페이스와 추상 클래스를 활용하여 유연한 코드구성을 할 수 있습니다.

10. 다음의 자바 코드 중 오류가 있는 것은?

① Power[] p = new Power[10];　　② Power p[] = new Power[10];
③ Power p[10] = new Power[10];　　④ Power[] p;

③ Power p[10] = new Power[10]; 이 오류가 있습니다.
배열을 선언하고 생성할 때, 배열의 크기는 반드시 변수로 지정해야 합니다.
배열의 크기를 변수로 지정하지 않고 고정된 크기를 직접 지정할 수 없습니다.
배열의 크기를 지정하려면 변수를 사용해야 합니다.

11. 다음 자바 코드에 대해 설명하는 문항 중 틀린 것은?

```
Book[] book = new Book[10];
```

① book은 배열에 대한 레퍼런스이다.
② Book 객체가 10개 만들어진다.
③ for (int i=0; i<book.length; i++) { book[i] = new Book(); } 로 객체들을 만들어야 비로소 배열이 완성된다.
④ book[0], book[1], book[2], ..., book[9] 모두 Book 객체에 대한 레퍼런스이다.

② Book 객체가 10개 만들어진다.가 틀린 설명입니다.
Book[] book = new Book[10];은 book이라는 배열을 생성하고, 배열의 크기를 10으로 지정합니다. new Book을 선언함으로 배열은 객체 생성이 아닌 객체 참조를 담는 공간으로 초기화시켜줍니다.

12. 다음 중 메소드 오버로딩에 실패한 사례는?

①
```
class A {
    int x;
    void f(int a) { x = a; }
    int f(int b) { return x+b; }
}
```

②
```
class A {
    int x;
    void f(int a) { x = a; }
    void f() { x = 0; }
}
```

③
```
class A {
    int x;
    int f() { return x; }
    int f(int a, int b) { return a+b; }
}
```

④
```
class A {
    static int x;
    static int f(int a) { return a+x; }
    static int f() { return 3; }
}
```

메소드 오버로딩은 같은 클래스 내에서 메소드 이름은 동일하지만 매개변수의 개수, 타입 또는 순서가 다른 경우에 적용이 됩니다.
①번의 경우에는 메소드의 시그니처(매개변수의 개수, 타입, 순서)가 동일한 f(int) 메소드와 f(int) 반환형을 가진 메소드가 중복되기 때문입니다. 메소드 오버로딩에서는 매개변수의 타입과 개수만 다르고 반환형은 같은 메소드로 구현할 수 없습니다.

13. 다음의 설명에 빈 곳에 적절한 단어를 채워라.

자바에서 상속받는 클래스를 _____ 라고 부르며, _____ 키워드를 이용하여 상속을 선언한다. 상속받은 클래스에서 상속해준 클래스의 멤버를 접근할 때 _____ 키워드를 이용한다. 한편, 객체가 어떤 클래스의 타입인지 알아내기 위해서는 _____ 연산자를 이용하며, 인터페이스로 클래스를 구현할 때는 _____ 키워드를 이용하여 구현한다.

자바에서 상속받는 클래스를 "하위 클래스" 라고 부르며, "extends" 키워드를 이용하여 상속을 선언한다. 상속받은 클래스에서 상속해준 클래스의 멤버를 접근할 때 "super" 키워드를 이용한다. 한편, 객체가 어떤 클래스의 타입인지 알아내기 위해서는 "instanceof" 연산자를 이용하며, 인터페이스로 클래스를 구현할 때는 "implements" 키워드를 이용하여 구현한다.

[14~ 15] 다음 자바 코드에 대해 답하라.

```
public class TV {
    int size;
    String manufacturer;

    public TV() {
        size = 32;
        manufacturer = "LG";
        System.out.println(size + "인치 " + manufacturer);
    }
    public TV(String manufacturer) {
        this.size = 32;
        this.manufacturer = manufacturer;
        System.out.println(size + "인치 " + manufacturer);
    }
    public TV(int size, String manufacturer) {
        this.size = size;
        this.manufacturer = manufacturer;
        System.out.println(size + "인치 " + manufacturer);
    }
}
```

14. main() 함수에서 new TV();와 new TV("삼성");를 실행하면 실행 결과는 각각 무엇인가?

new TV();를 실행하면 다음과 같은 결과가 출력됩니다:
32인치 LG

new TV("삼성");를 실행하면 다음과 같은 결과가 출력됩니다:
32인치 삼성

15. 65인치 "삼성" TV 객체를 생성하는 코드를 적어라.

TV tv = new TV(65, "삼성");
이 코드를 사용하면 위 조건의 객체가 생성되어 65인치 삼성 출력값이 나옵니다.

16. 다음 자바 클래스에는 컴파일 오류가 있다. 오류 부분을 지적하고 오류를 수정할 수 있는 방법을 모두 제시하라. 그리고 그 중 객체 지향 프로그래밍에 가장 적합한 방법을 설명하라.

```
class Person {
    private int age;
}

public class Example {
    public static void main(String[] args) {
        Person aPerson = new Person();
        aPerson.age = 17;
    }
}
```

두 가지의 방법이 존재합니다.

1) age 변수에 접근할 수 있는 public 메소드를 Person 클래스에 추가합니다. 이렇게 하면 외부에서 age 값을 설정하고 가져올 수 있습니다.
2) age 변수를 public으로 변경합니다. 이렇게 하면 외부에서 직접 aPerson.age와 같은 방식으로 접근할 수 있습니다.

객체 지향 프로그래밍의 관점에서 가장 적합한 방법은 첫 번째 방법입니다. 캡슐화가 가능하다는 객체 지향 프로그래밍 원칙에 따라 멤버 변수를 private으로 선언하고, 접근을 위한 public 메소드를 제공하는 것이 좋습니다. 이렇게 하면 외부에서는 객체의 내부 구조에 직접 접근하지 않고, 메소드를 통해 간접적으로 접근할 수 있으므로 객체의 무결성과 유지보수성을 높일 수 있습니다.

[17 ~ 18] 다음 자바 코드를 객체 지향 프로그래밍 관점에서 바람직한 코드로 수정하라.

```
class Power {
    public int kick;
    public int punch;
}
```

```
public class Example {
    public static void main(String[] args) {
        Power robot = new Power(10, 20); //(1)
        robot.setKick(30);  //(2)
        robot.setPunch(20);  //(2)
    }
}
```

객체 지향 프로그래밍 관점에서 바람직한 코드로 수정하기 위해서는 아래같이 코드를 작성할 수 있습니다.

```
class Power {
    private int kick;
    private int punch;
```

```
    public Power(int kick, int punch) {
        this.kick = kick;
        this.punch = punch;
    }

    public int getKick() {
        return kick;
    }

    public void setKick(int kick) {
        this.kick = kick;
    }

    public int getPunch() {
        return punch;
    }

    public void setPunch(int punch) {
        this.punch = punch;
    }
}

public class Example {
    public static void main(String[] args) {
        Power robot = new Power(10, 20); //(1)
        robot.setKick(30); //(2)
        robot.setPunch(20); //(2)

        int kickPower = robot.getKick();
        int punchPower = robot.getPunch();

        System.out.println("Kick power: " + kickPower);
        System.out.println("Punch power: " + punchPower);
    }
}
```

private int형태로 캡슐화를 사용합니다.

17. 위의 코드에서 (1)의 명령문을 수행하도록 생성자를 작성하라.

Power 클래스에 생성자를 **public Power로** 추가하여 int kick과 int punch를 매개변수로 받아 초기화합니다. 생성자를 통해 객체를 생성할 때 초기 값을 지정할 수 있습니다

18. 위의 코드에서 (2)의 명령문을 수행하도고 설정자(setter)메소드를 추가하라.

getter와 setter를 이용하여 setKick()와 setPunch() 메소드를 추가하여 외부에서 kick과 punch 값을 설정하여 **kickPower값과 punchPower값이 출력이 되도록 할 수 있습니다.**

19. 다음 코드에 대한 설명으로 틀린 것은?

```
ArrayList<Integer> v = new ArrayList<Integer>(30);
```

① Integer 타입 객체만 저장할 수 있는 구체화된 ArrayList를 생성하는 코드이다.
② ArrayList v는 원소를 30개만 저장할 수 있는 벡터이다.
③ v.add(10)을 호출하여 정수 10을 ArrayList에 삽입할 수 있다.
④ ArrayList v에는 실수 값을 삽입할 수 없다.

주어진 코드는 원소 타입으로 Integer를 갖는 ArrayList를 생성하는 코드입니다. 그러나 생성자의 매개변수인 30은 초기 용량을 나타내는 것이며, 실제로 저장 가능한 원소의 개수를 제한하는 것이 아닙니다. ArrayList는 필요에 따라 동적으로 크기를 조정할 수 있는 자료구조가 ArrayList입니다.
ArrayList v에는 원소의 개수에 제한이 없으며, 어떤 타입의 원소든지 저장할 수 있습니다. 따라서 ②의 설명이 틀린 내용입니다.

20. 객체지향의 특성 3가지를 적으시오.
객체 지향의 특성 3가지는 아래와 같습니다.
캡슐화 (Encapsulation): 관련 있는 데이터와 메서드를 하나의 단위로 묶어 캡슐화하는 것을 의미합니다. 캡슐화는 데이터와 메서드를 외부에서 접근할 수 있는 인터페이스를 제공하면서 내부 구현을 숨기고 정보 은닉을 실현합니다. 이를 통해 코드의 재사용성과 유지보수성을 높일 수 있습니다.

상속 (Inheritance): 기존의 클래스를 재사용하여 새로운 클래스를 작성하는 것을 의미합니다. 상속을 통해 부모 클래스의 특성과 동작을 자식 클래스가 물려받아 재사용할 수 있습니다. 이를 통해 코드 중복을 피하고 클래스 간의 계층 구조를 형성할 수 있습니다.

다형성 (Polymorphism): 동일한 메서드를 서로 다른 방식으로 동작하게 만드는 기능을 의미합니다. 다형성은 오버로딩(Overloading)과 오버라이딩(Overriding)을 통해 구현됩니다. 다형성을 활용하면 하나의 메서드나 클래스가 다양한 상황에서 다르게 동작할 수 있으며, 코드의 유연성과 확장성을 제공합니다.