1. 인터페이스의 특징에 대해 기술하시오.
2. 생성자(Constructor)란 무엇인지 서술하시오.
3. 객체지향프로그래밍(OOP)의 캡슐화에 대하여 서술하시오.
4. 객체지향프로그래밍(OOP)의 상속에 대하여 서술하시오.
5. 객체지향프로그래밍(OOP)의 다형성에 대하여 서술하시오.
6. 필드에 사용할 수 있는 접근제한자의 접근 가능 범위에 대해 간략하게 서술하시오.
7. 인터페이스와 추상클래스의 차이점에 대해 기술하시오.
8. 변수의 자료형 중 참조 자료형(Reference Type)에 대해 서술하시오.
9. 자바 프로그래밍 언어의 특징을 3가지 이상 기술하시오.
10. 문자열 "1"을 Wrapper클래스를 이용하여 parsing하시오.
String str = "1";

byte b =
short s =
int i =
long l =
float f =
double d =
boolean bl =
char c =

1. 인터페이스는 메서드를 선언만 할 수 있고 구현은 할 수 없는 것입니다. 단 java 8 이상부터는 default method 를 통해 구현할 수 있는 길이 생겼습니다. 인터페이스 같은 경우 final static 맴버 변수만을 가질 수 있으며 일반 맴버 변수는 가질 수 없습니다. 설계적 관점에서 클래스 같은 경우는 '존재' 에 관심을 가진다면 인터페이스 같은 경우는 '행동' 에 관심을 가지게 됩니다. 즉 인터페이스를 사용한다는 것은 이 구현체가 어떠한 존재인지보다 어떠한 행동을 하는지에 더 관심을 가진다고 보면 됩니다. 인터페이스 같은 경우 그 자체로 객체를 생성할 수 없으며, 인터페이스를 구현한 것을 통해 객체를 생성해야 합니다. 일반 클래스는 여러 인터페이스를 구현할 수 있으며 인터페이스는 다른 인터페이스를 상속할 수 있습니다.
2. 생성자는 JVM 에서 객체를 생성할 때에 호출하는 special method 입니다. 객체를 생성할 때에 클래스가 가지고 있는 맴버 변수를 초기화 해야 할 일이 있는데 이때 생성자를 사용하게 됩니다. 만약 클래스가 어떠한 생성자도 구현하지 않았다면 자동으로 default constructor 가 만들어지게 됩니다. 클래스가 어떤 클래스를 상속하게 된다면 생성자 호출을 할 때 가장 먼저 super class 의 constructor 가 호출되며 만약 생성자 구현에서 super() 함수를 호출하지 않았다면 super class 의 기본 생성자가 호출되게 됩니다. 생성자는 반드시 조상(Object) 가 먼저 호출이 되어야 하며, 그 다음에 자식이 차례대로 호출되게 됩니다.
3. 캡슐화 같은 경우는 정보를 숨기는 것입니다. 정보를 숨기는 가장 대표적인 방법은 맴버 변수를 private 로 선언하는 것입니다. 이렇게 함으로써 외부 객체(클래스) 같은 경우는 private 맴버 변수를 직접적으로 보거나 편집할 수 있는 방법이 없게 됩니다. 가장 일반적인 구현은 POJO 구현입니다. 즉 맴버변수는 private 로 두고 getter/setter 를 public 으로 두는 방식입니다. 단 캡슐화 같은 경우 맴버변수를 private 로 둔다고 해서 무조건 encapsulation 이 되는것이 아닙니다. 만약에 getter 에다가 객체의 참조 변수를 return 하게 된다면 이는 return 된 객체를 변형할 여지를 주는 것입니다.
4. 상속은 부모 클래스와 자식 클래스 관계를 만드는 것입니다. 자식 클래스 같은 경우 부모 클래스의 맴버 변수(단 private 변수는 상속 받지 않습니다.) 와 맴버 함수(단

private 함수는 상속 받지 않습니다.) 를 물려받는 것입니다. 맴버 변수같은 경우는 Override 를 할 수 없으며 만약에 자식 클래스에서 부모 클래스의 변수의 시그니처와 똑같은 변수를 만들게 된다면 이는 자식 클래스에 독립적으로 존재하는 또 다른 변수가 됩니다. 함수 같은 경우는 Override 를 할 수 있는데 Override 를 하려면 부모 클래스 함수의 시그니처와 동일해야 합니다. 즉 반환형, 함수 이름, 매개변수가 동일해야 합니다. 접근자 같은 경우 달라질수도 있지만 이는 확장되는 방향으로 가야 합니다. 즉 부모 클래스에서 protected 로 선언한 메서드를 자식 클래스에서 public 메서드로 override 할 수 있지만 private 메서도르 override 할수 없습니다. 리스코프 치환원칙을 지키면서 상속 관계를 만드는 것이 좋습니다.

5. 다형성 같은 경우는 같은 메서드 호출에 대하여 다르게 행동하는 것을 말합니다. 이는 상속 관계에서 나타나며 부모 클래스 A 에 draw 메서드가 있다 하고 B, C 가 A 를 상속받으면서 draw 메서드를 override 한다고 하면, A b = new B(); b.draw(); ,A c = new C(); c.draw() 를 할 때 동일한 함수 draw 를 호출하지만 실제 행동은 다른 것으로 알수 있습니다. 이는 자바 같은 경우 virtual method 방식으로 작동해서 생기는 것입니다.
6. private 는 오직 클래스 내부에서만 접근이 가능합니다.
   default 는 private 가 접근 가능한 것에 같은 패키지 내에 있는 클래스에 대해서 접근이 가능합니다.
   protected 는 default 에서 접근 가능한 것에 상속 관계에 있는 부모 클래스에 대해서 접근이 가능합니다.
   public 은 어떤 곳에서든 접근이 가능합니다.
7. 인터페이스 같은 경우는 일반 맴버 변수를 가지는 것이 불가능 합니다. 하지만 추상 클래스 같은 경우는 일반 맴버 변수를 가지는 것이 가능합니다. 또한 클래스는 인터페이스를 여러개 구현할 수 있는데, 추상 클래스 같은 경우는 하나 밖에 상속을 받지 못한다는 점이 있습니다.
8. 참조 자료형 같은 경우는 객체, 배열을 가리키는 변수입니다. 참조 자료형 변수는 일종의 포인터 개념이며, 참조 자료형 변수에 객체 그 자체가 담긴다기 보다는 그 객체에 대한 위치 정보(메모리 address) 가 담긴다고 보면 됩니다.
9. 자바 프로그래밍 언어 특성 같은 경우는 객체 지향, GC를 통한 메모리 자동 관리, 클래스 중심 구현, Write Once Run Anyway 등으로 정리할 수 있습니다.
10. byte b = Byte.parseByte(str);
    short s = Short.parseShort(str);
    int i = Integer.parseInt(str);
    long l = Long.parseLong(str);
    float f = Float.parseFloat(str);
    double d = Double.parseDouble(str);
    boolean bl = Boolean.parseBoolean(str); // 단 이 메서드 같은 경우는 오류가 납니다.
    char c = str.charAt(0); // Character 클래스 같은 경우 별도의 parse 함수가 없습니다.