1. 면접

2. 공부

[자바]

1. 개발 툴(IDE)

- Eclipse

- NetBeans(오라클)

- IntelliJ

2. 개발 환경(웹)(DB)

- 윈도우 기반

- JDK 1.8, JAVA8, JAVA2

- 이클립스

- (웹서버: 아파치/톰캣)

- (오라클 11g-10g-9i)

3. 자바 개발 환경

- JDK : 개발도구

- JRE : 실행환경

- JVM : 버추얼 머신, 가상 머신 **

- 컴파일

- 빌드 : 컴파일 + 부가작업

- 기계어

- 중간언어

- 가비지 컬렉터(자바가 다른 언어와 차이점)

- 포인터 없음(자바가 다른 언어와 차이점)

4. 개인 PC -> 개발 환경 설치

5. 객체 지향 프로그래밍

6. 자료형

- 값형 vs 참조형

- 값을 저장할 때 메모리에 직접 값을 저장하는 것이 값형, 어딘가에 값을 저장하고 그 주소를 저장하는 것이 참조형

- 왜? 값을 수정할 때, 메모리의 크기 변화 때문

7. 형변환

- 암시적인 형변환(안전한 형변환 원고지 1칸 -> 2칸) vs 명시적인 형변환(무조건 형변환 표현 원고지 2칸 -> 1칸)

- 값형 형변환 vs 참조형 형변환

8. 연산자

9. 메소드

- 메소드가 뭐냐?

- 코드 분리

- 호출 -> 코드 재사용(코드 단위)

- 인자, 반환값

- Call by Value(값형) vs Call by Reference(참조형)

10. 메소드 오버로딩? *******

- 인자값의 개수, 타입을 다르게 하여 이름이 같은 메소드를 여러 개 만드는 것

- 이유? 개발자 입장에서 성격이 비슷한 메소드를 접근하기 쉽고 외우기 쉽게하기 위해서

11. 메소드 오버라이딩?

- 상속

- 부모가 물려준 메소드를 시그니처는 그대로 쓰되, 그 안의 구현부는 자식이 다르게 쓸 수 있게 만드는 것(재정의)

- 이유? 부모의 메소드를 용도에 따라 개량을 하여 쓰기 위해서

12. 제어문 *********

- 코딩

13. 문자열

- 자바는 유니코드 사용 -> 문자 2바이트

- String vs StringBuffer

- 자바는 문자열이 불변이다.

    -> 잦은 문자열 변경 X

    -> 덩치 큰 문자열 변경 X

- String은 문자열이 불변이기 때문에 잦은 문자열을 변경하거나, 덩치 큰 문자열을 변경할 때 가비지가 많이 생겨 메모리를 많이 잡아먹고,

    프로그램이 느려질 수 있기 때문에 그 문제점을 해결하기 위해 StringBuffer를 쓴다.

- StringBuffer는 내부의 캐릭터 배열을 쓴다.

14. 배열

- 배열은? 같은 자료형의 변수를 연속으로 잡아놓은 공간

- 왜? 같은 자료형의 같은 ****** (학생들의 국어 점수를 처리할 때, 배열을 만들어 놓고 반복문 루프를 통해 업무를 처리하면 편리)

- 배열의 길이는 불변

- 코딩 ******

15. 클래스 **********(많이 설명)

- 객체 생성, 코드 단위, 생산성(코드 재사용성), 붕어빵

- 클래스 vs 객체 vs 인스턴스

16. 접근 지정자

- 멤버에만 지정(변수, 메소드)

- private : 클래스 외부에서 클래스 내부를 접근하지 못하게 막아놓는 접근 지정자

- public : 100% 접근 가능함

- protected : 상속 관계에 있는 자식에서만 접근할 수 있는 접근 지정자

17. 캡슐화

18. 생성자

- 객체가 생성될 때 가장 먼저 실행되는 메소드

- 왜? 멤버를 초기화하기 위한 전용 메소드, 코드 분리

- 생성자 오버로딩? 객체를 처음 만들 때, 다양한 모습으로 만들기 위해서 -> 객체의 다형성을 지원

19. 상속 **********

- 코드 재사용

- 부모 클래스의 멤버를 자식이 물려받음

- 기반이 되는 클래스를 가지고 확장할 때 쓴다.
- 상속(파생/확장)
- Object 클래스? 자바에서의 최상위 클래스(루트 클래스)
- 형 변환
- 오버라이딩
- 접근 지정자

20. 추상 클래스/메소드 *********
- 추상화? -> 단순화시키는 작업(사용자 경험을 살리기 위한 인터페이스 작업)

21. 인터페이스 *******
- 추상 클래스 vs 인터페이스 vs 일반 클래스
- 일반 클래스는 구현부만 있음 인터페이스는 추상 메소드만 추상 클래스는 일반 멤버도 가질 수 있음

22. final
- 변수(*), 메소드, 클래스
- 변수에 붙이면 값을 못 고치고, 메소드에 붙이면 오버라이딩을 못하고, 클래스에 붙이면 상속을 못받음
- 프로그램 안정성

23. 참조형 형변환
- 왜? 예제보고 정리하기 ***
- 업캐스팅? 다운캐스팅? *****
- 업캐스팅 : 서로 다른 자료형을 하나의 배열에 넣기 위해서(일괄 처리)
- 다운캐스팅 : 범용적으로 루프를 돌리다가 특정 타입만이 갖는 기능을 써야할 때 다운 캐스팅

24. 제네릭

- 클래스에서 멤버의 타입을 미리 결정X -> 객체를 생성할 때 타입을 결정O

- ArrayList<String>

- ArrayList<Integer>

- 왜? 생산성

25. 열거형, enum

- 가독성

- 주관식 -> 객관식

- 데이터를 좀 더 쉽게 구분하기 위해서 씀

- "red" -> Color.RED

26. 예외처리

- try 절

- catch 절

- finally 절

27. 컬렉션

- 예제(문제)*******

- 배열 vs 컬렉션

- 길이를 고정인게 배열, 컬렉션은 길이가 가변이고 가용성을 높인 배열임

- ArrayList(각각의 요소를 인덱스로 접근, 일괄 처리 - 스칼라배열) vs HashMap(각각의 요소를 키로 접근, 가독성 - 연관배열)

- Set(List와 유사) : List(중복 값 허용), Set(중복 값 불가)

*** 제어문, 배열(컬렉션) 문제

손 코딩!

[Oracle]

- 각 구성 요소의 의미 파악? 언제? 왜? -> 프로젝트 어떤 업무?

- ex) 뷰는? 자주쓰는 셀렉트 쿼리를 저장해놓는, 가독성↑

1. DB, DBMS, RDBMS

- DB : 데이터 집합

- DBMS : DB 관리 시스템

- RDBMS : 관계형 DB 관리 시스템 ↔ 비관계형은? txt파일, 빅데이터

2. SQL

- 구조화된 질의언어, 오라클(DB)과 대화를 하면서 데이터를 넣거나 가져올 수 있게하는 언어

- DDL(DB설계, 구조화), DML(데이터 조작), DCL(계정관리, 보안관리)?

3. char, varchar ↔ nchar, nvarchar

4. select, insert, update, delete

5. 테이블 스키마(Table Scheme) -> select 문제

-> where, order by, join 등..

6. between, in, like(_,%), null(name is null), distinct

7. 집계함수 + group by

- max, min, average, sum, count

8. 함수 - 개인적 연습

9. 설계

-> 프로젝트 DB 설계 해봤는지?

-> PK, FK, 관계?

-> not null, unique(PK(not null, not duplicate) - not null)

-> 무결성 유지?

10. 서브 쿼리

- 연습********, 질문

11. 조인**********

- 조인? 관계가 맺어진 두 테이블 이상을 합쳐 하나의 결과셋을 얻어내기 위해
- 고객 테이블 vs 주문내역 테이블
- inner join? 두 테이블에 동시에 존재하는 레코드를 가져오는 것
- outer join? 두 테이블에 동시에 존재하는 레코드뿐 아니라 나머지도 가져오는 것

12. union

- join vs union

13. View

- 질의를 저장하는 객체(테이블 복사(X))

- 역할? 왜?

14. 페이징 쿼리 구현 문제(서브 쿼리, rownum)

15. ERD, Entity, Attribute, Tuple, Relationship

- 의미? 역할? 등..

16. 정규화

*** 각종 select문 연습

[JDBC]

1. JDBC란?

- 응용 프로그램 ↔ 데이터 소스(DB) : 중간 계층

- 추상화 제공(인터페이스 제공)

- ODBC, OLEDB, ADO, ADO.NET

2. Connection, Statement, ResultSet

- 입출력 프로그램 짜기

3. Statement vs PreparedStatement

- Statement : 텍스트 쿼리, 정적 쿼리, 매번 컴파일, 인자값(유효성 처리 직접)

"select * from member"

- PreparedStatement : ?(인자값), 동적 쿼리, 컴파일 1번, 인자값(유효성 처리 자동)

"select * from member where seq=?"

4. DBCP, Database Connection Pool

- Connection 객체를 관리하는 기법

- 연결에 관한 비용을 줄이기 위해 Pool이라는 공간에 미리 저장해두고 불러와서 사용

[Semi Project]

1. 프로젝트 주제?

2. 담당 업무?(DB + Java) : 기술 소개서(필수)

3. 느낀점(하기 전 vs 하고난 후)

[HTML/CSS/Javascript/jQuery]

1. 프로젝트 결과

2. 코딩

3. 일

4. 클라이언트 환경

- 크롬 기반(크로스 브라우저, 웹 표준)

- HTML5

- Javascript Library(Framework) : jQuery

- UI Framework : Bootstrap

[Servlet & JSP]

1. 환경

- 웹서버(Apache)

- WAS, Web Application Server(Tomcat)

    -> Tomcat(Servlet Container, JSP Container)

    : 제우스, 웹로직, 웹스피어

- DB

2. Servlet 개념?

- 동적으로 클라이언트가 원하는 페이지 생성, 자바 기반

3. JSP 개념?

- 동적으로 클라이언트가 원하는 페이지 생성, HTML 기반

4. Servlet + JSP