|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **과목명** | **총문항수** | **O/X문제형** | **4지선다형** | **5지선다형** | **단답형** | **서술형** |
| JAVA(필기테스트) | 20문항 | 0문항 | 10문항 | 0문항 | 10문항 | 0문항 |

■ 대구분 : Java API

■ 소구분 : Object class/String class/StringBuffer/Wrapper (단답형)

■ 난이도 : 중

**[Q1]** 다음 프로그램은 간단한 회원 정보를 포함하고 있는 클래스를 작성한 것이다. 실행 결과를 적으시오.

**public** **class** UserTest {

 **public** **static** **void** main(String[] argv) {

 **new** UserTest().compare();

 }

 **private** **void** compare() {

 User a = **new** User("J", "Lee");

 User b = **new** User("J", "Lee");

 User c = a;

 System.*out*.println(a == b);

 System.*out*.println(a == c);

 System.*out*.println(a.equals(b));

 }

 **public** **class** User {

 **private** String firstName;

 **private** String lastName;

 **public** User(String firstName, String lastName) {

 **this**.firstName = **new** String(firstName);

 **this**.lastName = **new** String(lastName);

 }

 **public** **boolean** equals(User other) {

 **return** match(firstName, other.firstName)

 || match(lastName, other.lastName);

 }

 **private** **boolean** match(String part1, String part2) {

 **return** part1 == part2 && part1.equals(part2);

 }

 }

}

■ 대구분 : 기초문법

■ 소구분 : 조건문/반복문/연산자 (단답형)

■ 난이도 : 하

**[Q2]** 다음 반복 테스트(loop test) 클래스의 실행 결과를 적으시오.

**public** **class** ForLoop {

 **public** **static** **void** main(String[] argv) {

 **int** total = 0;

 **for**( **int** i=1; i<=5; i++ ) {

 **for**( **int** j=1; j<=i; i++ ) {

 total ++;

 }

 }

 System.*out*.println( total );

 }

}

■ 대구분 : 객체와 Class

■ 소구분 : 객체 생성과 사용 / Class선언 / 생성자 / 접근제한자 / 기타제한자(Static, Final, Abstract) (객관식)

■ 난이도 : 하

**[Q3]** 다음 코드 상의 \_\_\_ 위치에 들어갈 수 없는 키워드를 고르시오.

**\_\_\_** **class** ATestClass {

 **public** **static** **void** main(String[] argv) {

 }

}

1. public
2. private
3. abstract
4. final

■ 대구분 : 예외처리

■ 소구분 : Exception 카테고리/처리 방법  try-catch 및 throws (단답형)

■ 난이도 : 상

**[Q4]** 다음은 프로그래밍 도중 자주 사용되는 File 입출력(Input / Output) 예제이다. 텍스트 파일을 읽어 들여 내용을 화면에 출력하고 있다. 아래 코드 중 예외 처리 부분의 리소스 유출(resource leak) 가능성을 확인하고 올바른 코드를 적으시오.

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.IOException;

**import** java.io.InputStream;

**public** **class** PrintFile {

 **public** **static** **void** main(String[] argv) {

 **try** {

 **new** PrintFile().print("/tmp/log");

 } **catch** (Exception e) {

 e.printStackTrace();

 }

 }

 **public** **void** print(String filePath) **throws** IOException {

 File inputFile = **new** File(filePath);

 InputStream inputStream = **new** FileInputStream(inputFile);

 **try** {

 **while** (inputStream.available() > 0) {

 System.*out*.print((**char**) inputStream.read());

 }

 } **catch** (IOException e) {

 inputStream.close();

 }

 }

}

■ 대구분 : 예외처리

■ 소구분 : Exception 카테고리/처리 방법  try-catch 및 throws (객관식)

■ 난이도 : 중

**[Q5]** 다음은 수치 데이터(numeric data)를 문자열로 입력 받은 후, 정수형(integer type)으로 반환하는 예제이다. 잘못된 데이터 입력을 방지하기 위해 예외 처리 기능을 포함시켰다. 올바른 실행 결과를 고르시오.

**public** **class** HandleException {

 **public** **static** **void** main(String[] argv) {

 String number;

 System.*out*.println( *toInt*(number) );

 number = "9,900";

 System.*out*.println( *toInt*(number) );

 number = "10";

 System.*out*.println( *toInt*(number) );

 }

 **public** **static** **int** toInt(String strValue) {

 **if**( strValue == **null** || strValue.length() == 0 ) {

 **throw** **new** IllegalArgumentException();

 }

 **int** intValue = 0;

 **try** {

 intValue = Integer.*parseInt*(strValue);

 }

 **catch**(Exception e) {

 intValue = 0;

 }

 **return** intValue;

 }

}

① 실행 시 오류(runtime error)가 발생된다.

② 0 9900 10

③ 9900 10

④ 10

■ 대구분 : Data Type

■ 소구분 : Java data / Primitive / Reference type / Call by value (단답형)

■ 난이도 : 중

**[Q6]** 아래 프로그램의 출력 결과를 적으시오.

**public** **class** SetValues {

 **public** **static** **void** main(String[] argv) {

 String stringObj = "Hello";

 **int** intValue = 0;

 Float floatObj = **new** Float(1.0);

 *setValues*(stringObj, intValue, floatObj);

 System.*out*.println( stringObj + ", "

 + intValue + ", " + floatObj );

 }

 **private** **static** **void** setValues(String strValue, **int** intValue, Float floatObj) {

 strValue.replace("H", "h");

 strValue += " World";

 intValue = 99;

 floatObj.*valueOf*((**float**) 2.0);

 }

}

■ 대구분 : 객체와 클래스

■ 소구분 : 소구분 : 객체 생성과 사용 / Class선언 / 생성자접근제한자 / 기타제한자(Static, Final, Abstract) (단답형)

■ 난이도 : 상

[Q7] 아래 프로그램의 예상되는 출력 결과를 적으시오.

**public** **class** CodeBlocks {

 **private** **static** String *aField* = "";

 **private** String bField = "";

 **static** {

 *aField* += "A";

 }

 {

 *aField* += "B";

 bField += "B";

 }

 **public** CodeBlocks() {

 *aField* += "C";

 bField += "C";

 }

 **public** **void** append() {

 *aField* += "D";

 bField += "D";

 }

 **public** **static** **void** main(String[] argv) {

 System.*out*.println(*aField*);

 CodeBlocks cb = **new** CodeBlocks();

 cb.append();

 System.*out*.println(*aField*);

 System.*out*.println(cb.bField);

 }

}

■ 대구분 : Data Type

■ 소구분 : Java data / Primitive / Reference type / Call by value (객관식)

■ 난이도 : 하

**[Q8]** 자바 데이터 타입에 대한 설명 중 틀린 것을 고르시오.

① String 타입은 객체형이다.

② char 타입 변수는 2 byte의 메모리 공간을 사용한다.

③ char 타입 변수 선언 시 기본 값을 지정하지 않으면 ‘\u00000’ 값이 설정된다.

④ boolean 타입은 1 byte의 메모리 공간을 사용한다.

⑤ int 타입 변수는 4 byte의 메모리 공간을 사용한다.

■ 대구분 : 배열과 컬렉션

■ 소구분 : 배열의 활용 / 컬렉션의 활용 / generics / 배열과 컬렉션/Package (단답형)

■ 난이도 : 중

**[Q9]** 다음 프로그램은 자바의 컬렉션 API 예제이다. 올바른 출력 결과를 기술하시오.

**public** **class** Collection {

 List aList = **new** ArrayList();

 Set aSet = **new** HashSet();

 **public** **static** **void** main(String[] argv) {

 Collection collection = **new** Collection();

 collection.test();

 collection.print();

 }

 **private** **void** test() {

 **int** value = 1;

 addValue(value);

 addValue(value);

 value++;

 addValue(value);

 value++;

 addValue(value);

 }

 **private** **void** addValue(**int** value) {

 aList.add(value);

 aSet.add(value);

 }

 **private** **void** print() {

 **for**(Object obj : aList) {

 System.*out*.print(obj + " ");

 }

 System.*out*.println();

 **for**(Object obj : aSet) {

 System.*out*.print(obj + " ");

 }

 }

}

■ 대구분 : 객체와 Class

■ 소구분 : 객체 생성과 사용 / Class선언 / 생성자 / 접근제한자 / 기타제한자(Static, Final, Abstract) (객관식)

■ 난이도 : 중

**[Q10]** 아래 예제 프로그램은 컴파일 시 오류가 발생한다. 코드 중에서 문법 오류가 발생하는 라인을 고르시오.

**public** **class** AboutPerson {

 **public** **static** **void** main(String[] argv) {

① **static** **int** argvCount = argv.length;

② Person a = **new** Person();

③ a.setAge(10);

④ a.setName("Alice");

⑤ Person b = **new** Person();

⑥ b.age = 10;

⑦ a.name = "John";

 *print*(a);

 *print*(b);

 }

 **private** **static** **void** print(Person p) {

 System.*out*.println( "name : " + p.name + ", age : " + p.age );

 }

 **class** Person {

 **private** String name;

 **private** **int** age;

 **public** Person() {

 }

 **public** Person(**int** age) {

 **this**.age = age;

 }

 **public** String getName() {

 **return** name;

 }

 **public** String setName(String name) {

 **return** **this**.name;

 }

 **public** **int** getAge() {

 **return** age;

 }

 **public** **void** setAge(**int** age) {

 **this**.age = age;

 }

 }

}

■ 대구분 : OOP개념

■ 소구분 : Encapsulation/Inheritance/Polymorphism/Overloading/Overriding/Type Casting (객관식)

■ 난이도 : 하

**[Q11]** 다음 자바 언어에서의 객체지향 기법에 대한 설명 중 틀린 것을 모두 고르시오.

① 클래스는 복수의 인터페이스를 구현(implements)할 수 있다.

② 클래스는 복수의 부모 클래스(super class)로부터 상속(inherit) 받을 수 없다.

③ 인터페이스는 또 다른 인터페이스로부터 상속 받을 수 있다.

④ 모든 클래스는 상속을 통한 확장(extend)이 가능하다.

⑤ 인터페이스와 상위 클래스(super class)는 코드 재사용을 위한 기법이 아니다.

■ 대구분 : OOP개념

■ 소구분 : Encapsulation/Inheritance/Polymorphism/Overloading/Overriding/Type Casting (단답형)

■ 난이도 : 중

**[Q12]** 다음 프로그램의 실행 결과를 적으시오.

**public** **class** OverloadTest {

 **public** **static** **void** main(String[] argv) {

 **new** OverloadTest().test();

 }

 **private** **void** test() {

 Fruits fruits = **new** Apple();

 System.*out*.println(fruits.getName());

 }

 **public** **class** Fruits {

 **private** String name;

 **public** Fruits() {

 **this**.name = "unknown";

 }

 **final** **public** String getName() {

 **return** name;

 }

 }

 **public** **class** Apple **extends** Fruits {

 **public** String getName() {

 **return** "Apple";

 }

 }

}

■ 대구분 : 메모리관리

■ 소구분 : Garbage collector 개념 및 동작원리 / Memory Leak 탐지 및 예방 (객관식)

■ 난이도 : 상

**[Q13]** 다음은 Garbage Collection 에 대한 설명이다. 보기 중 잘못된 설명을 고르시오.

① 자바 프로그램 실행 중 객체가 garbage collect 되지 않을 수 있다.

② finalize() 메소드를 오버라이딩하고 필요한 자원을 반납했을 경우, 모든 자원은 확실하게 반납된다.

③ garbage collection 이 객체의 완전한 소멸(destruction)을 의미하는 것은 아니다.

④ System.gc() 메소드를 호출하여 강제로 garbage collection을 실행해도 garbage collectio이 보장되는 것은 아니다.

■ 대구분 : 예외처리

■ 소구분 : Exception 카테고리/처리 방법  try-catch 및 throws (단답형)

■ 난이도 : 상

**[Q14]** 다음은 예외 처리 예제이다. 예상되는 실행 결과를 적으시오.

**import** java.io.IOException;

**public** **class** HandleException2 {

 **public** **static** **void** main(String[] argv) {

 **new** HandleException2().test();

 }

 **private** **void** test() {

 **try** {

 Person person = **new** Person("John", -10);

 } **catch** (IOException e) {

 System.*out*.println("Exception catched");

 } **finally** {

 System.*out*.println("execute finally block");

 }

 }

 **public** **class** Person {

 **private** String name;

 **private** **int** age;

 **public** Person(String name, **int** age) **throws** IOException {

 **if** (age < 0)

 **throw** **new** IllegalParamException("Invalid input age : " + age);

 }

 **public** String getName() {

 **return** name;

 }

 **public** **int** getAge() {

 **return** age;

 }

 }

 **public** **class** IllegalParamException **extends** RuntimeException {

 **public** IllegalParamException(String msg) {

 **super**(msg);

 }

 }

}

■ 대구분 : Data Type

■ 소구분 : Java data / Primitive / Reference type / Call by value (객관식)

■ 난이도 : 하

**[Q15]** 다음 코드 및 예측 결과 중 틀린 해석을 고르시오.

①  int a = 3.5; // 컴파일 오류가 발생한다.

② int a1 = 5; double a2 = (float)a1; // 정상 동작한다.

③ int a = 9 / 0; // 컴파일 오류가 발생한다.

④ Integer a = new Integer(2); Integer b = new Integer(2); System.out.println( a == b ); // false를 출력한다.

■ 대구분 : OOP개념

■ 소구분 : Encapsulation/Inheritance/Polymorphism/Overloading/Overriding/Type Casting(객)

■ 난이도 : 중

**[Q16]** 다음 설명 중 틀린 것을 고르시오. (객관식)

① 추상 클래스(abstract class)는 하나 이상의 추상 메소드(abstract method)를 포함하고 있어야 한다.

② 추상 클래스는 객체를 생성할 수 없다.

③ protected 메소드는 모든 하위 클래스에서 호출할 수 있다.

④ 인터페이스를 구현한 클래스는 인터페이스에 포함된 모든 메소드를 구현하지 않아도 된다.

⑤ 자바의 모든 클래스는 Object 클래스의 자식 클래스이다.

⑥ A 클래스의 b 메소드를 하위 클래스 C에서 오버로딩한 경우, 하위 C 클래스에서 상위 클래스의 b 메소드를 호출할 수 있다.

■ 대구분 : 기초문법

■ 소구분 : 조건문/반복문/연산자 (단답형)

■ 난이도 : 하

**[Q17]** 아래 프로그램 출력 결과를 적으시오.

**public** **class** OperatorTest {

 **public** **static** **void** main(String[] argv) {

 **int** a = 10;

 **boolean** b = **false**;

 **if** ((b == **true**) || (a++ == 10)) {

 System.*out*.println("Equal " + a);

 } **else** {

 System.*out*.println("Not equal! " + a);

 }

 }

}

■ 대구분 : OOP개념

■ 소구분 : Encapsulation/Inheritance/Polymorphism/Overloading/Overriding/Type Casting (객관식)

■ 난이도 : 중

**[Q18]** 다음은 메소드 오버라이딩 예제이다. 잘못된 설명을 고르시오.

**public** **class** OverrideTest {

 **public** **static** **void** main() {

 **new** OverrideTest().test();

 }

 **private** **void** test() {

 SuperClass a = **new** SubClass();

 a.doh(1);

 }

 **class** SuperClass {

 **public** **char** doh(**char** c) {

 System.*out*.println("doh(char)");

 **return** 'c';

 }

 **public** **float** doh(**float** f) {

 System.*out*.println("doh(float)");

 **return** 1.0f;

 }

 }

 **class** OtherClass { }

 **class** SubClass **extends** SuperClass {

 **public** **void** doh(OtherClass o) {

 System.*out*.println("doh(OtherClass)");

 }

 }

}

① SuperClass의 doh(char c) 와 doh(float f) 메소드는 오버로딩(overloading)된 메소드이다.

② SubClass의 doh(OtherClass o) 메소드는 오버라이딩(overriding)된 메소드가 아니다.

③ test() 메소드 내에서 a.doh(1) 라인에서 컴파일 오류가 발생한다.

④ SuperClass의 doh(char c), doh(float f) 메소드를 SubClass에서 사용할 수 있다.

■ 대구분 : OOP개념

■ 소구분 : Encapsulation/Inheritance/Polymorphism/Overloading/Overriding/Type Casting (단답형)

■ 난이도 : 중

**[Q19]** 다음 프로그램의 실행 결과를 적으시오. (만일 실행할 수 없다면 그 이유를 적으시오.)

**public** **class** Poliymorphism {

 **private** **void** f() {

 System.*out*.println("base class");

 }

 **public** **static** **void** main(String[] argv) {

 Poliymorphism po = **new** Derived();

 po.f();

 }

 **class** Derived **extends** Poliymorphism {

 **public** **void** f() {

 System.*out*.println("sub class");

 }

 }

}

■ 대구분 : Data Type

■ 소구분 : Java data / Primitive / Reference type / Call by value (객관식)

■ 난이도 : 하

**[Q20]** 다음 프로그램의 실행결과로 올바른 것은?

**public** **class** AutoBoxing {

 **public** **static** **void** main(String[] argv) {

 **int** idx = 0;

 **char**[] charArray = **new** **char**[10];

 charArray[idx++] = '0';

 charArray[idx++] = 65;

 charArray[idx++] = 'A' + 1;

 System.*out*.println( charArray );

 }

}

① 컴파일 오류가 발생한다.

② 실행 시 오류가 발생한다.

③ 0AB

④ 065B