**실습문제 – 중첩클래스, 예외처리 해답**

1. **중첩 멤버 클래스에 대한 설명으로 틀린 것은 무엇입니까? 4**
2. 인스턴스 멤버 클래스는 바깥 클래스의 객체가 있어야 사용될 수 있다.
3. 정적 멤버 클래스는 바깥 클래스의 객체가 없어도 사용될 수 있다.
4. 인스턴스 멤버 클래스는 내부에는 바깥 클래스의 모든 필드와 메소드를 사용 할 수 있다.
5. 정적 멤버 클래스는 내부에는 바깥 클래스의 인스턴스 필드를 사용할 수 있다.
6. **로컬 클래스에 대한 설명으로 틀린 것은 무엇입니까? 3**
7. 로컬 클래스는 메소드 내부에 선언된 클래스를 말한다.
8. 로컬 클래스는 바깥 클래스의 모든 필드와 메소드를 사용할 수 있다.
9. 로컬 클래스는 static 키워드를 이용해서 정적 클래스로 만들 수 있다.
10. Final 특성을 가진 매개 변수나 로컬 변수만 로컬 클래스 내부에서 사용할 수 있다.
11. **익명 객체에 대한 설명으로 틀린 것은 무엇입니까? 3**
12. 익명 객체는 클래스를 상속하거나 인터페이스를 구현해야만 생성될 수 있다.
13. 익명 객체는 필드, 매개변수, 로컬변수의 초기값으로 주로 사용된다.
14. 익명 객체에는 생성자를 선언할 수 있다.
15. 부모 클래스나 인터페이스에 선언된 필드와 메소드 이외에 다른 필드와 메소드를 선언할 수

있지만, 익명 객체 내부에서만 사용이 가능한다.

1. **다음과 같이 Car 클래스 내부에 Tire와 Engine이 멤버 클래스로 선언되어 있습니다.**

**Example에서 멤버 클래스의 객체를 생성하는 코드를 작성해보세요.**

|  |
| --- |
| class Car{  class Tire { ….. }  static class Engine{ ….. }  }  public class Example {  public static void main(String[] args) {  Car car=new Car();  Car.Tire tire= car.new Tire();  Car.Engine engine=new Car.Engine();  }} |

**5. Outer 클래스의 내부 클래스 Inner의 멤버변수 iv의 값을 출력하세요**

|  |
| --- |
| class Outer{  class Inner{  public int iv=100;  }}  public class ABC {  public static void main(String[] args) {  Outer o=new Outer();  Outer.Inner ii=o.new Inner();  System.out.println(ii.iv);  }  } |

**7. Outer 클래스의 내부 클래스 Inner의 멤버변수 iv의 값을 출력하세요**

|  |
| --- |
| class Outer{  static class Inner{  public int iv=100;  }  }  public class ABC {  public static void main(String[] args) {  Outer.Inner ii=new Outer.Inner();  System.out.println(ii.iv);  }  } |

1. **AnonymousExample 클래스의 실행 결과를 보고 Vechicle 인터페이스의 익명 구현 객체를**

**이용해서 필드, 로컬 변수의 초기값과 메소드의 매개값을 대입해 보세요.**

|  |
| --- |
| interface Vehicle{  public void run();  }  class Aonumous{  public Vehicle field=new Vehicle(){  @Override  public void run() {  System.out.println("자전거가 달립니다.");  }  };  public void method(){  Vehicle locVar=new Vehicle(){  @Override  public void run() {  System.out.println("승용차가 달립니다.");  }  };  locVar.run();  }  public void function(Vehicle v){  v.run();  }  }  public class Example {  public static void main(String[] args) {  Aonumous anony=new Aonumous();  anony.field.run();  anony.method();  anony.function(new Vehicle(){  @Override  public void run() {  System.out.println("트럭이 달립니다."  }  });  }  } |

[결과화면] 자전거가 달립니다.

승용차가 달립니다.

트럭이 달랍니다.

1. **다음 Chatting 클래스는 컴파일 에러가 발생합니다. 원인이 무엇입니까?**

|  |
| --- |
| class Chat{  void start() { }  void sendMessage(String message){ }  }  public class Chatting {  public void startChat(String chatId){  String nickName=null; // null 값이다.  nickName=chatId; // 에러 String nickname=chatId  Chat chat=new Chat(){  @Override  public void start(){  while(true){  String inputData="안녕하세요.";  String message="["+ nickName + "]" + inputData;  sendMessage(message);  }}};  chat.start();  }  } |

* **다음 괄호안에 알맞은 답을 적으세요. (10번, 11번)**

1. **실행 중에 오동작이나 결과에 악영향을 미치는 예상치 못한 오류를 (예외)라고 한다.**
2. **예외가 발생할 가능성 있는 코드는 (try{ }) 블록으로 싸고, 예외가 발생할 때 처리하는**

**코드는 catch{ } 블록으로 작성하고, 예외가 발생하든 안 하든 최종적으로 실행할 코드는**

**finally{ } 블록에 작성한다. finally{ } 블록은 생략이 가능한다.**

**12. 자바의 예외 클래스 상속 관계를 이다. 괄호 안에 알맞은 것을 넣으세요.**

Java.lang.Object Throwable ( Exception )

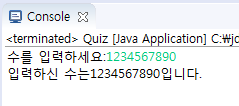
**13. 다음 중 예외 처리시 사용 되는 예약어가 아닌 것은? 3**

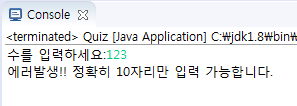
(1) catch (2) try (3) throw

**14. 다음의 문에서 statement2가 예외의 원인이라고 가정하여 보자**

|  |
| --- |
| try{  statement1;  statement2;  statement3;  }catch(Exception e) { 내용들.. }  }catch(Exception e) { 내용들.. }  }catch(Exception e) { 내용들.. }  }finally{  statement4;  }  statement5; |

***statement4 는 수행 되는가?*** *YES* ***statement5 는 수행 되는가?*** *YES*

**15. 다음 아래의 내용을 보시고 에러를 만들어보세요. (에러 클래스는 NumberFormatException)**



**16. 예외에 대한 설명 중 틀린 것은 무엇입니까? 3**

1. 예외는 사용자의 잘못된 조작, 개발자의 잘못된 코딩으로 인한 프로그램 오류를 말한다.
2. RuntimeException의 하위 예외는 컴파일러가 예외 처리 코드를 체크하지 않는다.
3. 예외는 try~catch 블록을 사용해서 처리된다.
4. 자바 표준 예외만 프로그램에서 처리할 수 있다.

**17. try~catch~finally 블록에 대한 설명 중 틀린 것은 무엇입니까? 3**

1. try { } 블록에는 예외가 발생할 수 있는 코드를 작성한다.
2. catch { } 블록은 try { } 블록에서 발생한 예외를 처리하는 블록이다.
3. try { } 블록에서 return문을 사용하면 finally { } 블록은 실행되지 않는다.

// retrun문을 만나면 finally 실행 후 빠져 나감

1. catch { } 블록은 예외의 종류별로 여러개를 작성할 수 있다.

**18. throws에 대한 설명으로 틀린 것은 무엇입니까?**

1. 생성자나 메소드의 선언 끝 부분에 사용되어 내부에서 발생된 예외를 떠넘긴다.
2. throws 뒤에는 떠 넘겨야 할 예외를 쉼표(,)로 구분해서 기술한다.
3. 모든 예외를 떠넘기기 위해 간단하게 throws Exception으로 작성할 수 있다.
4. 새로운 예외를 발생시키기 위해 사용된다. // throw

**19. throw에 대한 설명으로 틀린 것은 무엇입니까? 2**

1. 예외를 최초로 발생시키는 코드이다.
2. 예외를 호출한 곳으로 떠 넘기기 위해 메소드 선언부에 작성된다. throws
3. throw로 발생된 예외는 일반적으로 생성자 메소드 선언부에 throws로 떠넘겨진다.
4. throw 키워드 위에는 여러 객체 생성 코드가 온다.

**20. 다음과 같은 메소드가 있을 때 예외를 잘못 처리한 것은 무엇입니까? 3**

|  |
| --- |
| public void method() throws NumberFormatException, ClassNotFoundException{  ...............................................  } |

① try{ ② public void method2() thorws Exception{

method1(); method1();

}catch(Exception e) { } }

③ try{ ④ try{

method1(); method1();

} catch(Exception e){ } } catch(ClassNotFoundException e){ }

} catch(ClassNotFoundException e){ } } catch(NumberFormatException e){ }

**21. 다음 코드가 실행 되었을 때 출력 결과는 무엇입니까?**

|  |
| --- |
| public class Quiz {  public static void main(String[] args) {  String[] str={"10", "2a"};  int value=0;    for(int i=0;i<=2;i++){  try{  value=Integer.parseInt(str[i]);  }catch(ArrayIndexOutOfBoundsException e){  System.out.println("인텍스를 초과 했음");  }catch(NumberFormatException e){  System.out.println("숫자로 변환할 수 없음");  }finally{  System.out.println(value);  }  }  }  } |

0번지 : 10

1번지 : 숫자로 변환할 수 없다.

10

2번지 : 인텍스를 초과했음

10