6장. 클래스

1. 객체와 클래스에 대한 설명으로 틀린 것은 무엇입니까?

① 클래스는 객체를 생성하기 위한 설계도 (청사진)와 같은 것이다.
② new 연산자로 클래스의 생성자를 호출함으로써 객체가 생성된다.
③ 하나의 클래스로 하나의 객체만 생성할 수 있다.
④ 객체는 클래스의 인스턴스이다.

정답 : ③ 하나의 클래스로 하나의 객체만 생성할 수 있다.

2. 클래스의 구성 맴버가 아닌 것은 무엇입니까?

① 필드 (field)
② 생성자 (constructor)
③ 메소드 (method)
④ 로컬 변수 (local variable)

정답 : ④ 로컬 변수 (local variable)

3. 필드, 생성자, 메소드에 대한 설명으로 틀린 것은 무엇입니까?

① 필드는 객체의 데이터를 저장한다.
② 생성자는 객체의 초기화를 담당한다.
③ 메소드는 객체의 동작 부분으로, 실행 코드를 가지고 있는 블록이다.
④ 클래스는 반드시 필드와 메소드를 가져야 한다.

정답 : ④ 클래스는 반드시 필드와 메소드를 가져야 한다.

4. 필드에 대한 설명으로 틀린 것은 무엇입니까?

① 필드는 메소드에서 사용할 수 있다.
② 인스턴스 필드 초기화는 생성자에서 할 수 있다.

③ 필드는 반드시 생성자 선언 전에 선언되어야 한다.
④ 필드는 초기값을 주지 않더라도 기본값으로 자동 초기화된다.

정답 : ③ 필드는 반드시 생성자 선언 전에 선언되어야 한다.

5. 생성자에 대한 설명으로 틀린 것은 무엇입니까?

① 객체를 생성하려면 생성자 호출이 반드시 필요한 것은 아니다.
② 생성자는 다른 생성자를 호출하기 위해 this()를 사용할 수 있다.
③ 생성자가 선언되지 않으면 컴파일러가 기본 생성자를 추가한다.
④ 외부에서 객체를 생성할 수 없도록 생성자에 private 접근 제한자를 붙일 수 있다.

정답 : ① 객체를 생성하려면 생성자 호출이 반드시 필요한 것은 아니다.

6. 메소드에 대한 설명으로 틀린 것은 무엇입니까?

① 리턴값이 없는 메소드는 리턴 타입을 void로 해야 한다.
② 리턴 타입이 있는 메소드는 리턴값을 지정하기 위해 반드시 return문이 있어야 한다.
③ 매개값의 수를 모를 경우 "..."를 이용해서 매개 변수를 선언할 수 있다.
④ 메소드의 이름은 중복해서 선언할 수 없다.

정답 : ④ 메소드의 이름은 중복해서 선언할 수 없다.

7. 메소드 오버로딩에 대한 설명으로 틀린 것은 무엇입니까?

① 동일한 이름의 메소드를 여러 개 선언하는 것을 말한다.
② 반드시 리턴 타입이 달라야 한다.
③ 매개 변수의 타입, 수, 순서를 다르게 선언해야 한다.
④ 매개값의 타입 및 수에 따라 호출될 메소드가 선택된다.

정답 : ② 반드시 리턴 타입이 달라야 한다.

8. 인스턴스 멤버와 정적 멤버에 대한 설명으로 틀린 것은 무엇입니까?

① 정적 멤버는 static으로 선언된 필드와 메소드를 말한다.
② 인스턴스 필드는 생성자 및 정적 블록에서 초기화될 수 있다.
③ 정적 필드와 정적 메소드는 객체 생성 없이 클래스를 통해 접근할 수 있다.
④ 인스턴스 필드와 메소드는 객체를 생성하고 사용해야 한다.

정답 : ② 인스턴스 필드는 생성자 및 정적 블록에서 초기화될 수 있다.

9. final 필드와 상수(static final)에 대한 설명으로 틀린 것은 무엇입니까?

① final 필드와 상수는 초기값이 저장되면 값을 변경할 수 있다.
② final 필드와 상수는 생성자에서 초기화될 수 있다.
③ 상수의 이름은 대문자로 작성하는 것이 관례이다.
④ 상수는 객체 생성 없이 클래스를 통해 사용할 수 있다.

정답 : ② final 필드와 상수는 생성자에서 초기화될 수 있다.

10. 패키지에 대한 설명으로 틀린 것은 무엇입니까?

① 패키지는 클래스들을 그룹화시키는 기능을 한다.
② 클래스가 패키지에 소속되려면 패키지 선언을 반드시 해야 한다.
③ import문은 다른 패키지의 클래스를 사용할 때 필요하다.
④ mycompany 패키지에 소속된 클래스는 yourcompany에 옮겨 놓아도 동작한다.

정답 : ④ mycompany 패키지에 소속된 클래스는 yourcompany에 옮겨 놓아도 동작한다.

11. 접근 제한에 대한 설명으로 틀린 것은 무엇입니까?

① 접근 제한자는 클래스, 필드, 생성자, 메소드의 사용을 제한한다.
② public 접근 제한은 아무런 제한 없이 해당 요소를 사용할 수 있게 한다.

③ default 접근 제한은 해당 클래스 내부에서만 사용을 허가한다.
④ 외부에서 접근하지 못하도록 하려면 private 접근 제한을 해야 한다.

정답 : ③ default 접근 제한은 해당 클래스 내부에서만 사용을 허가한다.

12. 다음 클래스에서 해당 멤버가 필드, 생성자, 메소드 중 어떤 것인지 빈 칸을 채우세요.

```
public class Member {
  private String name; ---> (          )

  public member(String name) { ... } ---> (          )

  public void setName(String name) { ... } ---> (          )
}
```

정답 : 필드(Field), 생성자(Constructor), 메소드(Method)

13. 현실 세계의 회원을 Member 클래스로 모델링하려고 합니다. 회원의 데이터로는 이름, 아이디, 패스워드, 나이가 있습니다. 이 데이터들을 가지는 Member 클래스를 선언해보세요.

데이터 이름	필드 이름	타입
이름	name	문자열
아이디	id	문자열
패스워드	password	문자열
나이	age	정수

[Member.java]

```
public class Member {
  // 작성 위치
}
```

정답

```
public class Member {
	String name;
	String id;
	String password;
	int age;
}
```

실행 결과

없음

14. 위에서 작성한 Member 클래스에 생성자를 추가하려고 합니다. 다음과 같이 Member 객체를 생성할 때 name 필드와 id 필드를 외부에서 받은 값으로 초기화하려면 생성자를 어떻게 선언해야 합니까?

```
Member user1 = new Member("홍길동", "hong");
Member user2 = new Member("강자바", "java");
```

[Member.java]

```
public class Member {
  // 작성 위치
}
```

정답

```
public class Member {
	String name;
	String id;
	String password;
	int age;

	Member(String name, String id) {
		this.name = name;
		this.id = id;
	}
}
```

실행 결과

없음

15. MemberService 클래스에 login() 메소드와 logout() 메소드를 선언하려고 합니다. login() 메소드를 호출할 때는 매개값으로 id와 password를 제공하고, logout() 메소드는 id만 매개값으로 제공합니다. MemberService 클래스와 login(), logout() 메소드를 선언해보세요.

① login() 메소드는 매개값 id가 "hong", 매개값 password가 "12345"일 경우에만 true로 리턴하고 그 이외의 값일 경우에는 false를 리턴하도록 하세요.

② logout() 메소드의 내용은 "로그아웃 되었습니다."가 출력되도록 하세요.

리턴 타입	메소드 이름	매개 변수(타입)
boolean	login	id(String), password(String)
void	logout	id(String)

[MemberService.java]

```
public class MemberService {
  // 작성 위치
}
```

[MemberServiceExample.java]

```
public class MemberServiceExample {
  public static void main(String[] args) {
    MemberService memberService = new MemberService();
    boolean result = memberService.login("hong", "12345");
    if(result) {
      System.out.println("로그인 되었습니다.");
```

```
        memberService.logout("hong");
    } else {
        System.out.println("id 또는 password가 올바르지 않습니다.");
    }
  }
}
```

정답

```
public class MemberService {
        boolean login
        (String id, String password) {
                if(id.equals("hong") &&
                        password.equals("12345")) {
                                return true;
                } else {
                        return false;
                }
        }

        void logout(String id) {
                System.out.println
                ("로그아웃 되었습니다.");
        }
}

public class MemberServiceExample {

        public static void main(String[] args) {
                MemberService memberService
                = new MemberService();

                boolean result = memberService.login
                ("hong", "12345");

                if(result) {
                        System.out.println
                        ("로그인 되었습니다.");
                        memberService.logout("hong");
                } else {
                        System.out.println
                        ("id 또는 password가 "
                                        + "올바르지 않습니다.");
```

```
            }
        }
}
```

실행 결과

로그인 되었습니다.
로그아웃 되었습니다.

16. PrinterExample 클래스에서 Printer 객체를 생성하고 println() 메소드를 호출해서 매개값을 콘솔에 출력하려고 합니다. println() 메소드의 매개값으로는 int, boolean, double, string값을 줄 수 있습니다. Printer 클래스에서 println() 메소드를 선언해보세요.

[Printer.java]

```
public class Printer {
  // 작성 위치
}
```

[PrinterExample.java]

```
public class PrinterExample {
  public static void main(String[] args) {
    Printer printer = new Printer();
    printer.println(10);
    printer.println(true);
    printer.println(5.7);
    printer.println("홍길동");
  }
}
```

정답

```
public class Printer {
	void println(int value) {
		System.out.println(value);
	}

	void println(boolean value) {
		System.out.println(value);
	}

	void println(double value) {
		System.out.println(value);
	}

	void println(String value) {
		System.out.println(value);
	}
}

public class PrinterExample {

	public static void main(String[] args) {
		Printer printer = new Printer();
		printer.println(10);
		printer.println(true);
		printer.println(5.7);
		printer.println("홍길동");

	}

}
```

실행 결과

```
10
true
5.7
홍길동
```

17. 16번 문제에서는 Printer 객체를 생성하고 println() 메소드를 생성했습니다. Printer 객체를 생성하지 않고 PrinterExample 클래스에서 다음과 같이 호출하려면 Printer 클래스를 어떻게 수정하면 될까요?

[Printer.java]

```
public class Printer {
  // 작성 위치
}
```

[PrinterExample.java]

```
public class PrinterExample {
  public static void main(String[] args) {
    Printer printer = new Printer();
    printer.println(10);
    printer.println(true);
    printer.println(5.7);
    printer.println("홍길동");
  }
}
```

정답

```
public class Printer {
	static void println(int value) {
		System.out.println(value);
	}

	static void println(boolean value) {
		System.out.println(value);
	}

	static void println(double value) {
		System.out.println(value);
	}

	static void println(String value) {
		System.out.println(value);
	}
}

public class PrinterExample {
```

```
        public static void main(String[] args) {
                Printer.println(10);
                Printer.println(true);
                Printer.println(5.7);
                Printer.println("홍길동");

        }
}
```

실행 결과

```
10
true
5.7
홍길동
```

18. ShopService 객체를 싱글톤으로 만들고 싶습니다. shopServiceExample 클래스에서 ShopService의 getInstance() 메소드로 싱글톤을 얻을 수 있도록 ShopService 클래스를 작성해보세요.

[ShopService.java]

```
public class ShopService {
  // 작성 위치
}
```

[ShopServiceExample.java]

```
public class ShopServiceExample {
  public static void main(String[] args) {
    ShopService obj1 = ShopService.getInstance();
    ShopService obj2 = ShopService.getInstance();

    if(obj1 == obj2) {
      System.out.println("같은 ShopService 객체 입니다.");
```

```
    } else {
      System.out.println("다른 ShopService 객체 입니다.");
    }
  }
}
```

정답

```
public class ShopService {
	private static ShopService singleton
	= new ShopService();

	private ShopService() {}

	public static ShopService getInstance() {
		return singleton;
	}
}

public class ShopServiceExample {

	public static void main(String[] args) {
		ShopService obj1 =
		ShopService.getInstance();
		ShopService obj2 =
		ShopService.getInstance();

		if(obj1 == obj2) {
			System.out.println
			("같은 ShopService 객체 입니다.");
		} else {
			System.out.println
			("다른 ShopService 객체 입니다.");
		}

	}

}
```

실행 결과

```
같은 ShopService 객체 입니다.
```

19. 은행 계좌 객체인 Account 객체는 잔고(balance) 필드를 가지고 있습니다. balance 필드는 음수값이 될 수 없고, 최대 백만 원까지만 저장할 수 있습니다. 외부에서 balance 필드를 마음대로 변경하지 못하도록 하고, 0 <= balance <= 1,000,000 범위의 값만 가질 수 있도록 Account 클래스를 작성해 보세요.

① setter와 Getter를 이용하세요.
② 0과 1,000,000은 MIN_BALANCE와 MAX_BALANCE 상수를 선언해서 이용하세요.
③ Setter의 매개값이 음수이거나 백만 원을 초과하면 현재 balance 값을 유지하세요.

[Account.java]

```
public class Account {
  // 작성 위치
}
```

[AccountExample.java]

```
public class AccountExample {
  public static void main(String[] args) {
    Account account = new Account();

    account.setBalance(10000);
    System.out.println("현재 잔고 : " + account.getBalance());

    account.setBalance(-100);
    System.out.println("현재 잔고 : " + account.getBalance());

    account.setBalance(2000000);
    System.out.println("현재 잔고 : " + account.getBalance());

    account.setBalance(300000);
```

```
    System.out.println("현재 잔고 : " + account.getBalance());
  }
}
```

정답

```
public class Account {
	public static final int
	MIN_BALANCE = 0;
	public static final int
	MAX_BALANCE = 1000000;

	private int balance;

	public int getBalance() {
		return balance;
	}

	public void setBalance(int balance) {
		if
		(balance < Account.MIN_BALANCE
		|| balance > Account.MAX_BALANCE) {
			return;
		}
		this.balance = balance;
	}


}

public class AccountExample {

	public static void main(String[] args) {
		Account account = new Account();

		account.setBalance(10000);
		System.out.println
		("현재 잔고: " + account.getBalance());

		account.setBalance(-100);
		System.out.println
		("현재 잔고: " + account.getBalance());

		account.setBalance(2000000);
```

```
                System.out.println
                ("현재 잔고: " + account.getBalance());

                account.setBalance(300000);
                System.out.println
                ("현재 잔고: " + account.getBalance());

        }

}
```

실행 결과

```
현재 잔고: 10000
현재 잔고: 10000
현재 잔고: 10000
현재 잔고: 300000
```

20. 다음은 키보드로부터 계좌 정보를 입력받아서 계좌를 관리하는 프로그램입니다. 실행 결과를 보고 알맞게 BankApplication 클래스의 메소드를 작성해보세요.

[Account.java]

```
public class Account {
  private String ano;
  private String owner;
  private int balance;

  public Account(String ano, String owner, int balance) {
    this.ano = ano;
    this.owner = owner;
    this.balance = balance;
  }

  public String getAno() {
    return ano;
```

```
  }

  public void setAno(String ano) {
    this.ano = ano;
  }

  public String getOwner() {
    return owner;
  }

  public void setOwner(String owner) {
    this.owner = owner;
  }

  public int getBalance() {
    return balance;
  }

  public void setBalance(int balance) {
    this.balance = balance;
  }
}
```

[BankApplication.java]

```
import java.util.Scanner;


public class BackApplication {
  private static Account[] accountArray = new Account[100];
  private static Scanner scanner = new Scanner(System.in);

  public static void main(String[] args) {
    boolean run = true;
```

```
  whlie(run) {
    System.out.println("-----------------------------");
    System.out.println
    ("1.계좌생성 | 2.계좌목록 | 3.예금 | 4.출금 | 5.종료");
    System.out.println("-----------------------------");
    System.out.println("선택> ");

    int selectNo = scanner.nextInt();

    if(selectNo == 1) {
      createAccount();
    } else if(selectNo == 2) {
      accountList();
    } else if(selectNo == 3) {
      depositt();
    } else if(selectNo == 4) {
      widthdraw();
    } else if(selectNo == 5) {
      run = false;
    }
  }
  System.out.println("프로그램 종료");
}

// 계좌생성하기

private static void createAccount() {
  // 작성 위치
}

// 계좌목록보기

private static void accountList() {
  // 작성 위치
```

```
}

// 예금하기

private static void deposit() {
  // 작성 위치
}

// 출금하기

private static void widthdraw() {
  // 작성 위치
}

// Account 배열에서 ano와 동일한 Account 객체 찾기

private static Account findAccount(String ano) {
  // 작성 위치
}
```

정답

```
public class Account {

	private String ano;
	private String owner;
	private int balance;

	public Account
	(String ano, String owner, int balance) {
		this.ano = ano;
		this.owner = owner;
		this.balance = balance;
	}

	public String getAno() {
		return ano;
	}
```

```
	public void setAno(String ano) {
		this.ano = ano;
	}

	public String getOwner() {
		return owner;
	}

	public void setOwner(String owner) {
		this.owner = owner;
	}

	public int getBalance() {
		return balance;
	}

	public void setBalance(int balance) {
		this.balance = balance;
	}


}


import java.util.Scanner;

public class BankApplication {
	private static Account[] accountArray
	= new Account[100];
	private static Scanner scanner
	= new Scanner(System.in);

	public static void main(String[] args) {
		boolean run = true;
		while(run) {
			System.out.println
			("-----------------------------");
			System.out.println
		("1.계좌생성 | 2.계좌목록 | "
				+ "3.예금 | 4.출금 | 5.종료");
		System.out.println
		("-----------------------------");
		System.out.println("선택> ");

		int selectNo = scanner.nextInt();
```

```
            if(selectNo == 1) {
                createAccount();
            } else if(selectNo == 2) {
                accountList();
            } else if(selectNo == 3) {
                deposit();
            } else if(selectNo == 4) {
                withdraw();
            } else if(selectNo == 5) {
                run = false;
            }
        }

        System.out.println("프로그램 종료");

    }

    private static void withdraw() {
        System.out.println
        ("-----------------------------");
        System.out.println("출금");
        System.out.println
        ("-----------------------------");
        System.out.print("계좌번호: ");
        String ano = scanner.next();
        System.out.print("출금액: ");
        int money = scanner.nextInt();

        Account account = findAccount(ano);
        if(account == null) {
            System.out.println
            ("결과: 계좌가 없습니다.");
            return;
        }
        account.setBalance
        (account.getBalance() - money);
        System.out.println
        ("결과: 출금이 성공되었습니다.");

    }

    private static void deposit() {
        System.out.println
        ("-----------------------------");
        System.out.println("예금");
        System.out.println
```

```
        ("-----------------------------");
        System.out.print("계좌번호: ");
        String ano = scanner.next();
        System.out.print("예금액: ");
        int money = scanner.nextInt();

        Account account = findAccount(ano);
        if(account == null) {
                System.out.println
                ("결과: 계좌가 없습니다.");
                return;
        }
        account.setBalance
        (account.getBalance() + money);
        System.out.println
        ("결과: 입금이 성공되었습니다.");
}


private static Account findAccount
(String ano) {
        Account account = null;
        for
        (int i = 0; i < accountArray.length;
        i++) {
                if(accountArray[i] != null) {
                        String dbAno =
                        accountArray[i].getAno();
                        if(dbAno.equals(ano)) {
                                account = accountArray[i];
                                break;
                        }
                }
        }
        return account;
}

private static void accountList() {
        System.out.println
        ("-----------------------------");
        System.out.println("계좌목록");
        System.out.println
        ("-----------------------------");

        for
```

```
        (int i = 0; i < accountArray.length;
        i++) {
                Account account = accountArray[i];
                if(account != null) {
                        System.out.print
                        (account.getAno());
                        System.out.print("      ");
                        System.out.print
                        (account.getOwner());
                        System.out.print("      ");
                        System.out.print
                        (account.getBalance());
                        System.out.println();
                }
        }

}

private static void createAccount() {
        System.out.println
        ("---------------------------");
        System.out.println("계좌생성");
        System.out.println
        ("---------------------------");

        System.out.print("계좌번호:");
        String ano = scanner.next();

        System.out.print("계좌주:");
        String owner = scanner.next();

        System.out.print("초기입금액:");
        int balance = scanner.nextInt();

        Account newAccount
        = new Account(ano, owner, balance);

        for
        (int i = 0; i < accountArray.length;
        i++)
        {
                if(accountArray[i] == null) {
                        accountArray[i] = newAccount;
                        System.out.println
                        ("결과: 계좌가 "
                                        + "생성되었습니다.");
```

```
                    break;
                }
            }
        }

}
```

실행 결과

[계좌생성 실행결과]

```
-----------------------------
1.계좌생성 | 2.계좌목록 | 3.예금 | 4.출금 | 5.종료
-----------------------------
선택> 1

계좌생성
-----------------------------
계좌번호 : 111-111
계좌주: 홍길동
초기입금액: 10000
결과: 계좌가 생성되었습니다.
-----------------------------
1.계좌생성 | 2.계좌목록 | 3.예금 | 4.출금 | 5.종료
-----------------------------
선택> 1

계좌생성
-----------------------------
계좌번호 : 111-222
계좌주: 김자바
초기입금액: 20000
결과: 계좌가 생성되었습니다.
```

[계좌목록 실행결과]

```
----------------------------
1.계좌생성 | 2.계좌목록 | 3.예금 | 4.출금 | 5.종료
----------------------------
선택> 2

계좌목록
----------------------------
111-111 홍길동 10000
111-222 김자바 20000
```

[예금 실행결과]

```
----------------------------
1.계좌생성 | 2.계좌목록 | 3.예금 | 4.출금 | 5.종료
----------------------------
선택> 3

예금
----------------------------
계좌번호 : 111-111
예금액: 500
결과: 예금이 성공되었습니다.
```

[출금 실행결과]

```
----------------------------
1.계좌생성 | 2.계좌목록 | 3.예금 | 4.출금 | 5.종료
----------------------------
선택> 4

출금
----------------------------
```

계좌번호 : 111-222
출금액: 3000
결과: 출금이 성공되었습니다.

[계좌목록/종료 실행결과]

1.계좌생성 | 2.계좌목록 | 3.예금 | 4.출금 | 5.종료

선택> 2

계좌목록

111-111 홍길동 10000
111-222 김자바 20000

1.계좌생성 | 2.계좌목록 | 3.예금 | 4.출금 | 5.종료

선택> 5
프로그램 종료