Chapter7

1. 섯다카드 20장을 담는 SutdaCard배열을 초기화 하시오

```
for(int i = 0; i < cards.length; i++) {
  StudaCard sutda;
  if(10 < i) {
    i -= 10;
    sutda = new SutdaCard(i, false);
  } else {
    if(i == 1 || i == 3 || i == 8) {
      sutda = new SutdaCard(i, true);
    } else {
      sutda = new SutdaCard(i, false);
    }
  }
  cards[i] = sutda;
}
```

2. 1문제의 새로운 메소드를 추가하고 테스트

- shuffle - 카드 섞기

```
public void shuffle() {
  for(int i = 0; i < cards.length - 1; i++) {
    int index = (int)Math.random() * cards.length;
    SutdaCard tmp = cards[i];
    cards[i] = cards[index];
    cards[index] = tmp;
  }
}
```

- pick - 지정된 위치의 카드 반환

```
 public SutdaCard pick(int index) {
  return cards[index];
}
```

- pick - 임의의 위치의 SutdaCard를 반환한다.

```
public SutdaCard pick() {
  int index = (int)Math.random()*cards.length;
  return cards[index];
}
```

3. 오버라이딩의 정의와 필요성
    - 오버라이딩이란 조상의 메소드를 재정의 하는 것
        - 반환타입, 이름, 매개변수가 같아야한다.
    - 상속 받은 부모의 메소드에서 자식 클래스의 필요한 내용으로 수정할 수 있다.

4. 오버라이딩의 조건으로 옳지 않은 것.
    - d - 조성의 메소드보다 넓은 범위로만 변경할 수 있다.
    - e - 조상의 메소드보다 적거나 같은 수의 예외를 선언할 수 있다.

5. 컴파일 에러 파악하기
    - 부모의 생성자가 default 생성자가 없고 자식 클래스에서 부모의 default 생성자를 사용하기에 에러가 난다.
    - 부모 클래스에서 default 생성자를 생성해 주던가 자식 클래스에서 부모의 정의되어 있는 생성자를 사용한다.

6. 생성자를 호출해야하는 이유
    - 상속을 받는건 부모클래스가 자식 클래스에 추가 된다는 느낌?
    - 따라서 자식을 생성할 때 부모의 맴버변수, 메소드들도 생성 되어야한다.
    - 그렇기에 자식 클래스에서 부모 클래스의 생성자를 사용해야 한다.
    - 조상에 정의된 인스턴스 변수들이 초기화 되도록 하기 위해서

7. 생성자의 순서와 실행결과
    - child() → child(int x) → Parent() → Parent(int x)
    - 결과 → 200

- child 에서 this로 child(int x)가 실행되었고 child에서 보이지 않지만 super()를 실행 시킨다.
  super는 Parent()이고 Parent에서 this로 Parent(int x)를 실행시킨다.
- getX는 Parent에서 정의된 메소드이기에 Parent의 x가 리턴된다.

8. 접근제어자 접근범위가 넓은 것에서 좁은 순으로 나열
   - a public → protected → default → private

9. final을 붙일 수 있는 대상과 붙였을때 의미가 옳지 않은것
   - c - 오버라이딩을 할 수가 없다.

10. 외부에서 접근할 수 없도록 접근 제어자를 붙이고, getter, setter를 만들어라
    - private를 붙여준다.
    - lombok 라이브러리를 추가 후 @Getter @Setter 어노테이션을 붙여준다.
    - getter, setter 만들기 - 매개변수가 있는 메소드는 매개변수의 유효성 검사를 꼭 해주자

```
public void setIsPowerOn(boolean isPowerOn) {
  this.isPowerOn = isPowerOn;
}
public boolean getIsPowerOn() {
  return this.isPowerOn;
}
```

11. 10번에서 이전 체널로 이동하는 메소드를 추가하자.

```
private int prevChannel = 0;

public void setChannel(int channel) {
  this.prevChannel = this.channel;
  this.channel = channel;
}

public int gotoPrevChannel() {
  int tmp = this.channel;
  this.channel = this.prevChannel;
```

```
  this.prevChannel = tmp;
}
```

12. 접근 제어자에 대한 설명으로 옳지 않은 것은?
    - c - 지역변수에는 접근 제어자를 사용할 수 없다.
    - 접근 제어자가 사용될 수 있는 곳 - 클래스, 맴버변수, 메소드, 생성자

13. Math클래스의 생성자는 private인 이유
    - Math의 생성자가 public이 되면 보안성이 깨진다.
    - 생성자를 이용하여 멤버변수에 값을 임의로 넣고 수정이 가능해지기 때문이다.
    - 모든 메소드가 static메소드이고, 인스턴스변수가 존재하지 않는다.
    - 따라서 외부로의 불필요한 접근을 막기 위해 private으로 지정해준 것이다.

14. 섯다카드의 값이 변경되지 않는 값으로 수정
    - num과 isKwang에 final을 붙여준다?

15. 형변환을 올바르게 하지 않은 것은?
    - e → 부모 클래스를 자식 클래스로 형변환 할 수 없다.

16. instanceof 의 연산결과가 true가 아닌것
    - e - 엠뷸런스는 소방차와 같이 차를 상속받지만 다른 인스턴스이다.

17. 공통된 부분을 클래스로 만들어서 상속받는 코드로 수정

```
class Unit {
  int x, y;
  void move(int x, int y){ }
  void stop() { }
}

class Marine extends Unit {
```

```
  void stimPack() {}
}

class Tank extends Unit {
  void changeMode() { }
}

class Dropship extends Unit {
  void load() { }
  void unload() { }
}
```

18. 실행 결과 얻기

```
public void action(Robot r) {
  if(r instanceof DanceRobot) {
    DanceRobot d = (DanceRobot)r;
    d.dance();
  } else if( r instanceof SingRobot) {
    SingRobot s = (SingRobot)r;
    s.sing();
  } else {
    DrawRobot dr = (DrawRobot)r;
    dr.draw();
  }
}
```

19. 다음은 물건을 구입하는 사람을 정의한 클래스이다 이 클래스는 멤버변수 Buyer .
로 돈 과 장바구니 를 가지고 있다 제품을 구입하는 기능의 메서드와 장 (money) (cart)
. buy
바구니에 구입한 물건을 추가하는 메서드 구입한 물건의 목록과 사용금액 그리고 남
add , ,
은 금액을 출력하는 메서드를 완성하시오 summary

```
public void buy(Product p) {
  if(p.price <= money ) {
    money -= p.price;
    add(p);
  } else {
    return;
  }
}

public void add(Product p) {
  if(cart.length <= i) {
```

```
    Product[] cart2 = new Product[cart.length * 2];
    System.arraycopy(cart, 0, cart2, 0, cart.length);
    crat = cart2;
  }
    cart[i] = p;
    i++;
//////////////////////////
  for(int i = 0; i < cart.length; i++) {
    if(cart[i] == null) {
      cart[i] = p;
    } else {
      if(i == cart.length -1) {
        Product[] cart2 = new Product[cart.length * 2];
        for(int i = 0; i < cart.length; i++) {
          cart2[i] = cart[i];
        }
        cart2[cart.length] = p;
        cart = cart2;
      }
    }
  }
}

public void summary() {
  int result = 0;
  for(int i = 0; i < cart.length; i++;) {
    syso(cart[i])
    result += p.price;
  }
  사용 금액 result
  남은 금액 money
```

20. 코드의 실행 결과

```
100 200
```

21. 메소드의 매개변수로 가능한 것

```

```

22. shape클래스를 상속 받는 클래스 생성

```
public class Circle() extends Shape {
  ouble r;
```

```
  double calcArea() {
    return r * r * Math.PI;
  }
}

public class Rectangle() extends Shape {
  double width;
  double height;

  double calcArea() {
    return width * height;
  }

  public boolean isSquare() {

  }
}
```

23. 22에서 면적을 구하는 메소드를 작성

```
public double sumArea(Shape[] arr) {
  double sum = 0;
  for(int i = 0; i < arr.length; i++) {
    sum += arr[i].calcArea();
  }
  return sum;
}
```

24. 인터페이스의 장점이 아닌 것은

- e - 클래스간의 연결을 도와준다.

25. Outer 클래스의 내부 클래스 Inner의 멤버 변수 iv의 값을 출력하시오

```
Outer outer = new Outer();
outer.Inner inner = new outer.Inner();
inner.iv;
```

26. 25와 같고 Inner만 static으로 정의된 내용

```
Outer outer = new Outer();
System.out.println(outer.Inner.iv);
```

27. 올바른 실행결과 얻기

```

```

28. 익명클래스 변환

```
f.addWindowListener(new WindowAdpter() {
  public void windowClosing(WindowEvent e) {
    e.getWindow().setVisible(false);
    e.getWindow().dispose();
    System.exit(0);
  }
});
```

29. 지역 클래스에서 외부 클래스의 인스턴스 멤버와 static 멤버에 모두 접근할 수 있지만 지역변수는 final이 붙은 상수만 접근할 수 있는 이유 무엇인가?
    - 메소드가 수행을마쳐서 지역변수가 소멸된 시점에도 지역클래스의 인스턴스가 소멸된 지역변수를 참조하려는 경우가 발생할 수 읶기 때문이다.
    -