Oracle Direct Seminar

ORACLE®

Javaプログラマ資格ポイント解説

日本オラクル株式会社

Oracle Direct

以下の事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。以下の事項は、マテリアルやコード、機能を提供することをコミットメント（確約）するものではないため、購買決定を行う際の判断材料になさらないで下さい。オラクル製品に関して記載されている機能の開発、リリースおよび時期については、弊社の裁量により決定されます。

Agenda

- Oracle認定Java資格概要
- Javaプログラマ資格試験(OCJ-P)の紹介
- ポイント解説

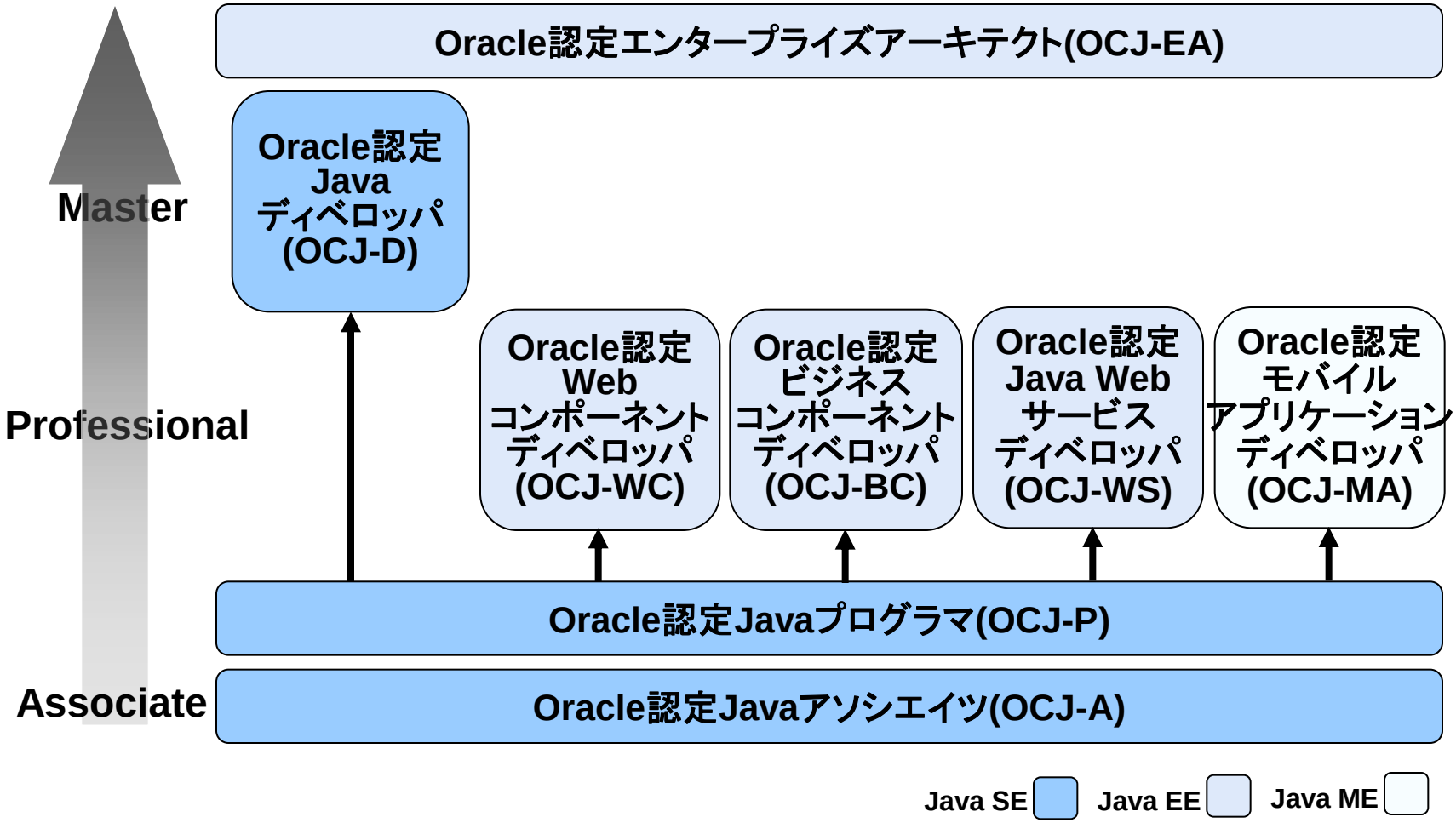
Agenda

- **Oracle認定Java資格概要**
- Javaプログラマ資格試験(OCJ-P)の紹介
- ポイント解説

Oracle Java認定資格

OCJ-A(旧SJC-A) Oracle認定Javaアソシエイツ	Javaテクノロジー概要/オブジェクト指向概要 /Javaプログラミング基礎
OCJ-P(旧SJC-P) **Oracle認定Javaプログラマ**	**Javaプログラミング基礎** **Java SE基本ライブラリ**
OCJ-D(旧SJC-D) Oracle認定Javaディベロッパ	Java SEを使用したアプリケーション作成
OCJ-WC(旧SJC-WC) Oracle認定Webコンポーネントディベロッパ	サーブレット/JSPに関する包括的な知識
OCJ-BC(旧SJC-BC) Oracle認定ビジネスコンポーネントディベロッパ	EJB全般とEJB関連サービスについての包括的な知識
OCJ-WS(旧SJC-WS) Oracle認定Java Webサービスディベロッパ	Webサービス関連知識
OCJ-MA(旧SJC-MA) Oracle認定モバイルアプリケーションディベロッパ	Java ME関連知識
OCJ-EA(旧SJC-EA) Oracle認定エンタプライズアーキテクト	Java EEアーキテクト関連知識

Oracle Java認定資格



Agenda

- Oracle認定Java資格概要
- **Javaプログラマ資格試験(OCJ-P)の紹介**
- ポイント解説

Oracle認定JavaプログラマSE 6 (OCJ-P)資格

Java Standard Edition 6 Programmer Certified Professional Exam

- Javaプログラミング言語の包括的なスキルを証明したいプログラマを対象とした資格

試験番号	CX-310-065
前提条件	なし
出題問題数	60問
合格ライン	58%
出題形式	多岐選択式/ドラッグ &ドロップ
対象バージョン	Java SE 6

- スキルレベル
  - Java言語の基本文法を理解している
  - Java言語を使用してOOプログラミングができる
  - Java.lang、java.util、java.ioパッケージのクラスの使用方法を理解し、プログラム作成ができる

OCJ-P SE 6出題範囲

- Section 1 : 宣言、初期化、スコープ
- Section 2 : フロー制御
- Section 3 : APIコンテンツ
- Section 4 : 並行性
- Section 5 : オブジェクト指向コンセプト
- Section 6 : コレクション/ジェネリックス
- Section 7 : Java言語の基礎

reference

- 認定資格情報
  http://www.oracle.com/jp/education/certification/middleware-172606-ja.html
- Oracle認定Javaプログラマ資格詳細
  http://education.oracle.com/pls/web_prod-plq-dad/db_pages.getpage?page_id=41&p_exam_id=1Z0_851&p_org_id=70&lang=JA
- トレーニングコース情報
  http://education.oracle.com/pls/web_prod-plq-dad/db_pages.getlppage?page_id=212&path=SJPF&p_org_id=70&lang=JA

Agenda

- Oracle認定Java資格概要
- Javaプログラマ資格(OCJ-P)の紹介
- **ポイント解説**

OCJ-P SE 6出題範囲

- **Section 1 : 宣言、初期化、スコープ**
- Section 2 : フロー制御
- Section 3 : APIコンテンツ
- Section 4 : 並行性
- Section 5 : オブジェクト指向コンセプト
- Section 6 : コレクション/ジェネリックス
- Section 7 : Java言語の基礎

Section 1: 宣言、初期化、スコープ
列挙型

- 関連する定数をまとめて宣言する型。Java言語では、定数を扱うクラスとなる。
- 宣言方法

  enum 列挙型名 { 要素1, 要素2, … }
- 使用方法

  列挙型名 参照変数名 = 列挙型名.要素名;
- 使用例

```
enum  Color { RED, BLUE, WHITE }
Color  c  = Color.RED;
System.out.println(Color.BLUE);
```

Section 1: 宣言、初期化、スコープ
列挙型

- switch文との組み合わせ

例:

```
enum Color { RED, BLUE, WHITE }
Color c = Color.RED;
switch(c) {
      case RED:
          System.out.print("赤");
          break;
      case BLUE:
          System.out.print("青");
          break;
      case WHITE:
          System.out.print("白");
          break;
}
```

Section 1: 宣言、初期化、スコープ
列挙型

- 定義パターン
  - あるクラスのメンバとして

```
class Sample {
    enum Color { RED, BLUE,WHITE }
    …
}
```

  - 別クラスとして

```
enum Color { RED, BLUE,WHITE}
class Sample {
    …
}
```

  - 別クラス＆別ソースファイルで

```
enum Color {RED, BLUE,WHITE}
```

```
class  Sample  {
    …
}
```

問題1:

列挙型の定義および使用として適切なコードはどれですか？

a.
```
public class Test {
    public static void main(String[] args) {
      enum Seasons { SPRING, SUMMER, AUTUMN, WINTER }
      System.out.println(Seasons.WINTER);
    }
  }
```
b.
```
public class Test {
    enum Seasons { SPRING, SUMMER, AUTUMN, WINTER }
    public static void main(String[] args) {
      Seasons s = new Seasons();
      System.out.println(s.WINTER);
    }
  }
```
c.
```
public class Test {
    enum Seasons { SPRING, SUMMER, AUTUMN, WINTER }
    public static void main(String[] args) {
      System.out.println(Seasons.WINTER);
    }
  }
```

解答1:

列挙型の定義および使用として適切なコードはどれですか？

a.
```
public class Test {
    public static void main(String[] args) {
      enum Seasons { SPRING, SUMMER, AUTUMN, WINTER }
      System.out.println(Seasons.WINTER);
    }
  }
```

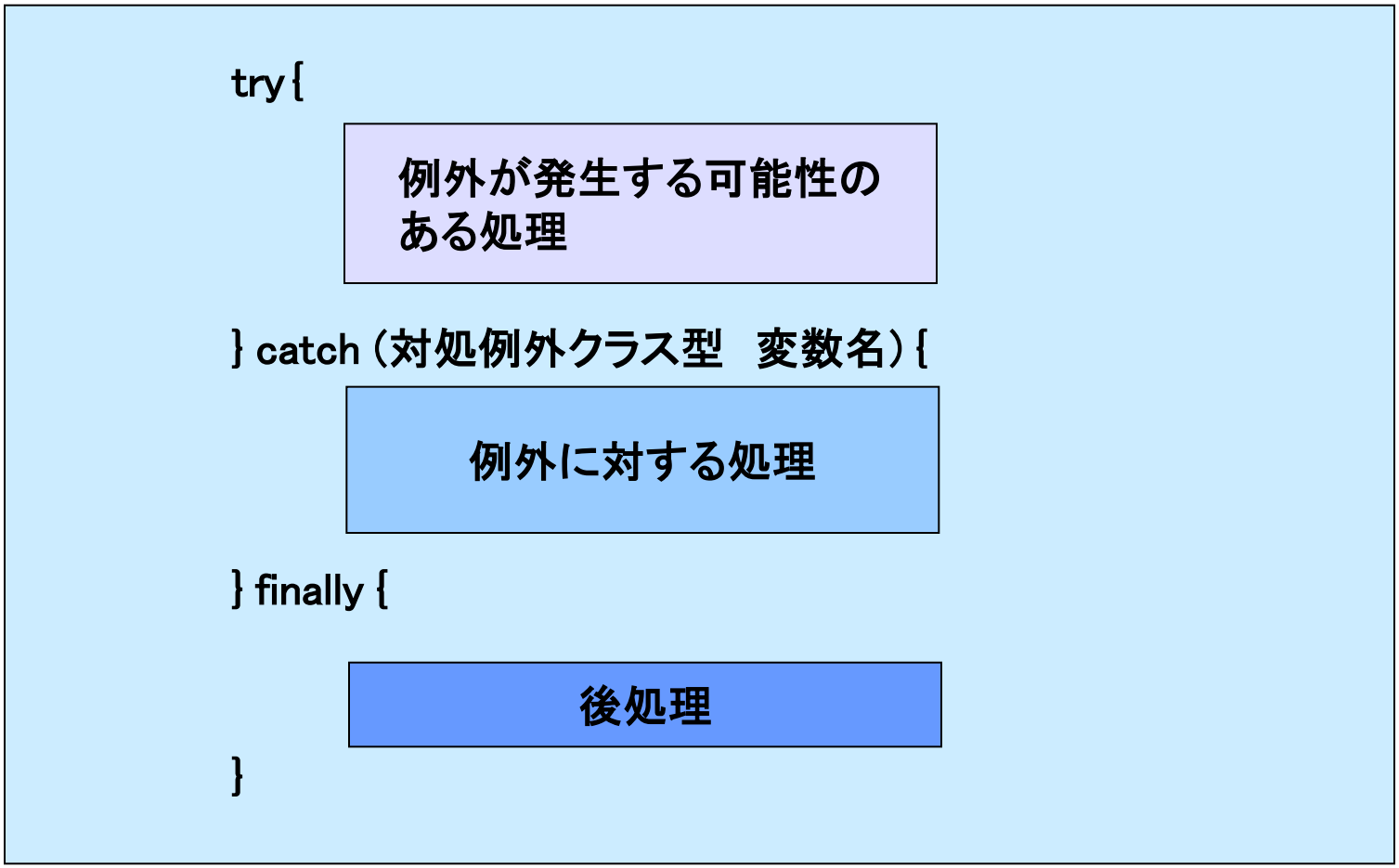
b.
```
public class Test {
    enum Seasons { SPRING, SUMMER, AUTUMN, WINTER }
    public static void main(String[] args) {
      Seasons s = new Seasons();
      System.out.println(s.WINTER);
    }
  }
```

c.
```
public class Test {
    enum Seasons { SPRING, SUMMER, AUTUMN, WINTER }
    public static void main(String[] args) {
      System.out.println(Seasons.WINTER);
    }
  }
```

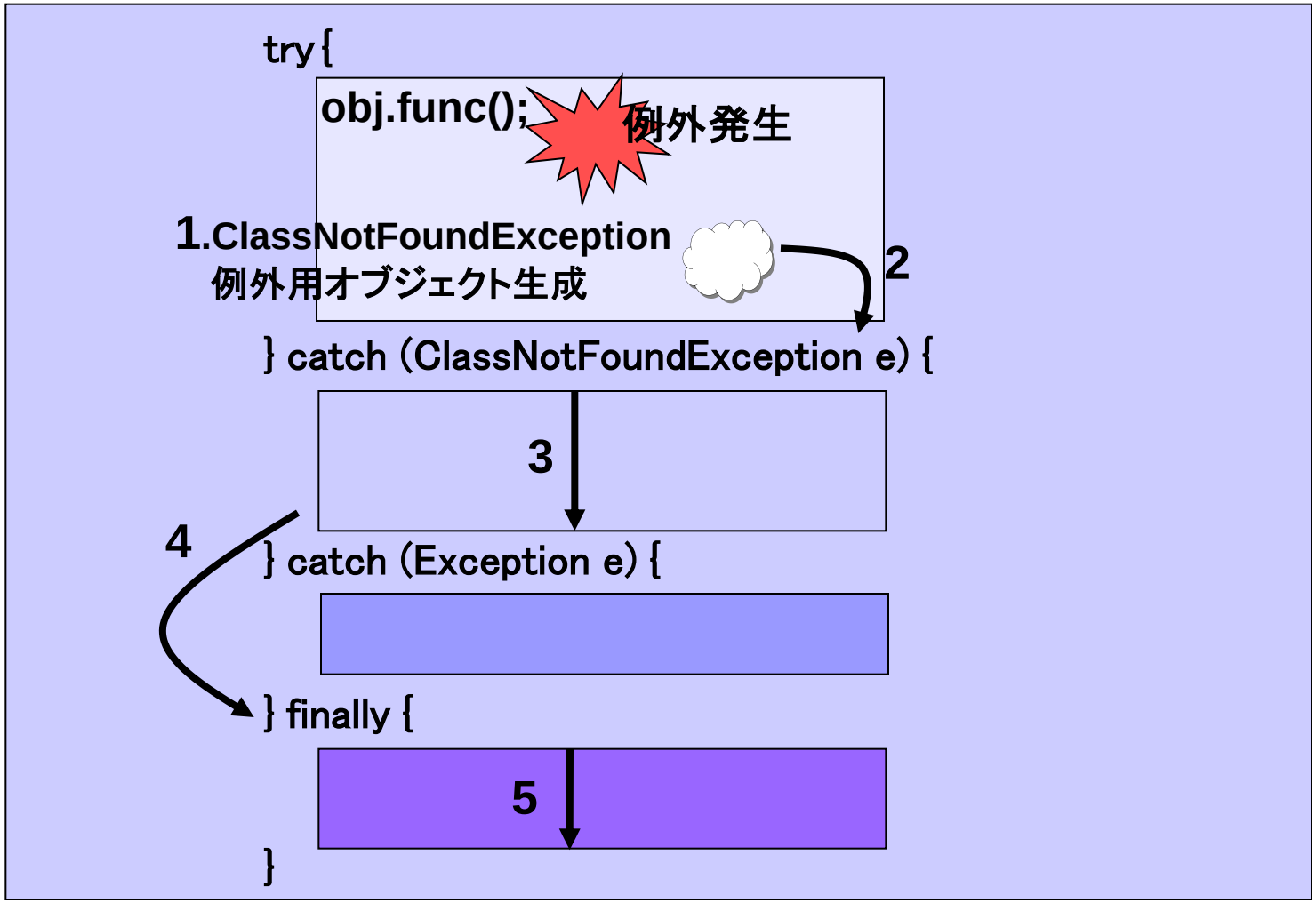
OCJ-P SE 6出題範囲

- Section 1 : 宣言、初期化、スコープ
- **Section 2 : フロー制御**
- Section 3 : APIコンテンツ
- Section 4 : 並行性
- Section 5 : オブジェクト指向コンセプト
- Section 6 : コレクション/ジェネリックス
- Section 7 : Java言語の基礎

Section 2: フロー制御
例外処理



Section 2: フロー制御
例外処理



問題2:

以下のコードがあります。実行結果はどれですか？

```
 1. class Birds {
 2.   public static void main(String [] args) {
 3.     try {
 4.       throw new Exception();
 5.     } catch (Exception e) {
 6.       try {
 7.         throw new Exception();
 8.       } catch (Exception e2) { System.out.print("inner "); }
 9.       System.out.print("middle ");
10.     }
11.     System.out.print("outer ");
12.   }
13. }
```

a. inner
b. inner outer
c. middle outer
d. inner middle outer
e. middle inner outer
f. コンパイルエラーになる
g. 実行時に例外がスローされる

解答2:

以下のコードがあります。実行結果はどれですか？

```
 1. class Birds {
 2.   public static void main(String [] args) {
 3.     try {
 4.       throw new Exception();
 5.     } catch (Exception e) {
 6.       try {
 7.         throw new Exception();
 8.       } catch (Exception e2) { System.out.print("inner "); }
 9.       System.out.print("middle ");
10.     }
11.     System.out.print("outer ");
12.   }
13. }
```

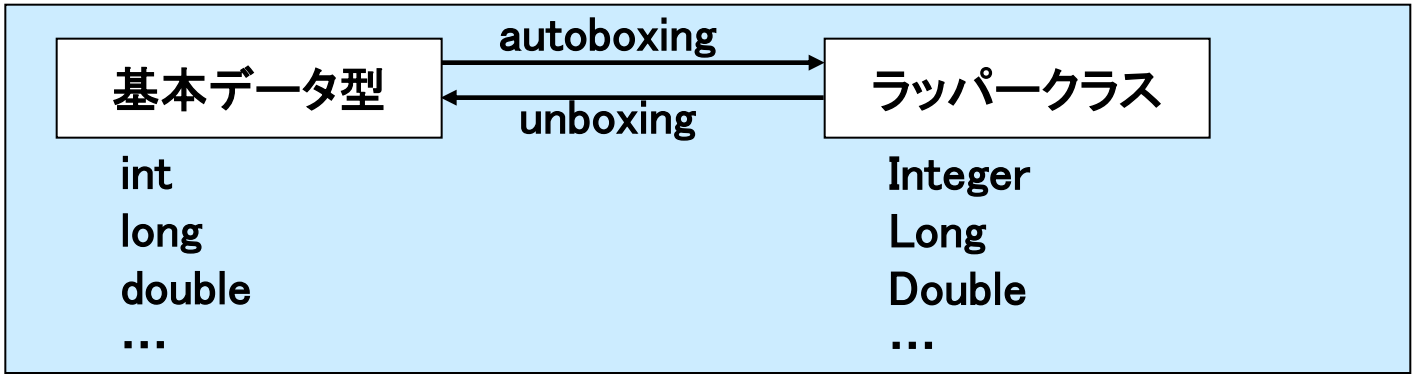
a. inner
b. inner outer
c. middle outer
d. inner middle outer
e. middle inner outer
f. コンパイルエラーになる
g. 実行時に例外がスローされる

OCJ-P SE 6出題範囲

- Section 1 : 宣言、初期化、スコープ
- Section 2 : フロー制御
- **Section 3 : APIコンテンツ**
- Section 4 : 並行性
- Section 5 : オブジェクト指向コンセプト
- Section 6 : コレクション/ジェネリックス
- Section 7 : Java言語の基礎

Section 3: APIコンテンツ
Autoboxing/Unboxing

- 基本データ型と対応ラッパークラス間で自動的に型変換



- 変換ルール
  - autoboxing
    基本データ型からその基本データ型に対応したラッパークラス型へ変換。その後参照型の型変換ルールに従う。
  - unboxing
    ラッパークラスのデータを対応した基本データ型に変換。その後基本データ型の型変換ルールに従う。

Section 3: APIコンテンツ
Autoboxing/Unboxing

例： autoboxing

```
int i = 10;
Integer iobj = i;
```

Integer型に変換されて代入される

```
Double dobj = 10.0f;
```

10.0はFloat型に変換後Double型に
代入することになるためエラー

例： unboxing

```
int i = new Integer(10);
```

int型に変換されて代入される

```
short s = new Integer(100);
```

int型の100に変換後、short型に
代入することになるためエラー

問題3:

以下のうち適切なコードはどれですか？(2つ選択してください。)

a. `float f = new Integer(100);`

b. `Short s = new Integer("100");`

c. `Double d = 3.14f;`

d. `short s = new Byte((byte)10);`

e. `Float f = 10.0;`

解答3:

以下のうち適切なコードはどれですか？（2つ選択してください。）

a. `float f = new Integer(100);`

b. `Short s = new Integer("100");`

c. `Double d = 3.14f;`

d. `short s = new Byte((byte)10);`

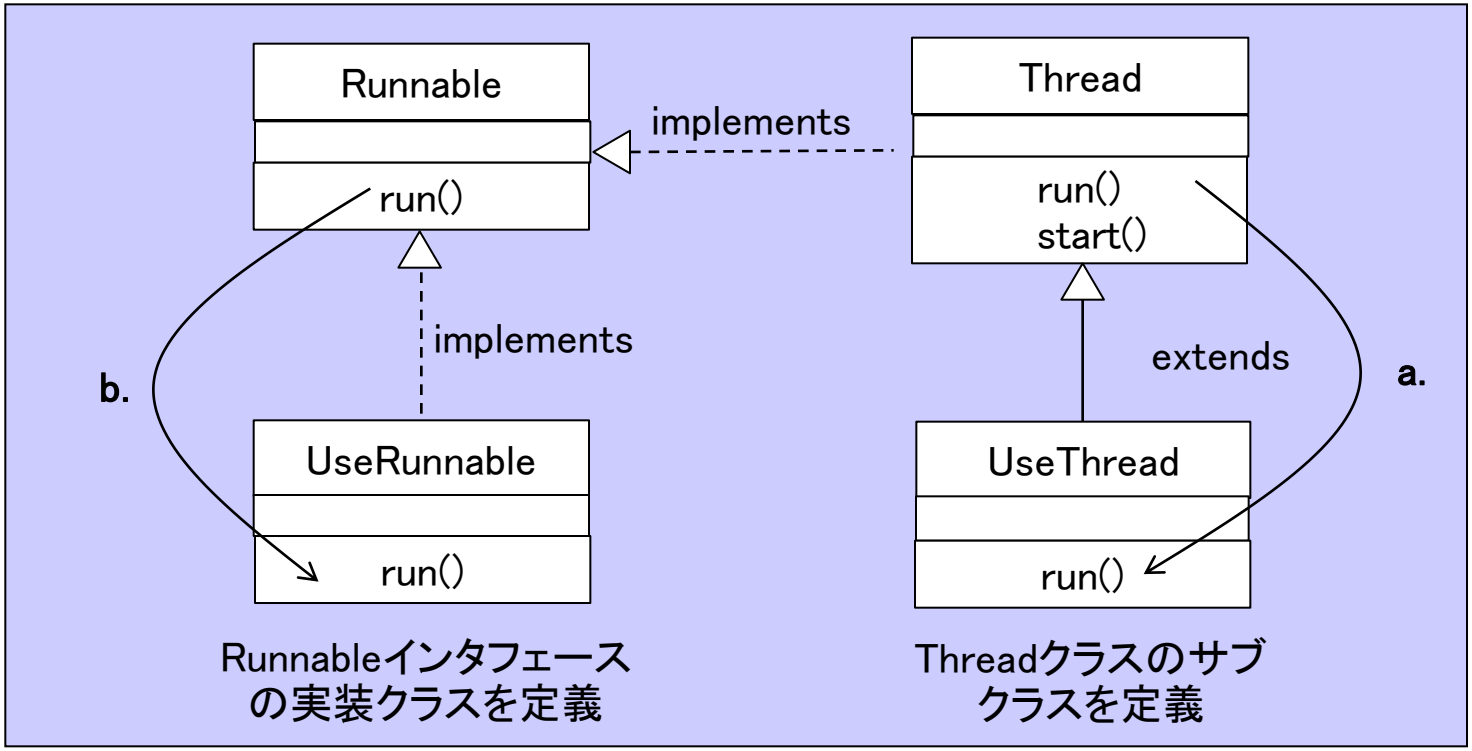
e. `Float f = 10.0;`

OCJ-P SE 6出題範囲

- Section 1 : 宣言、初期化、スコープ
- Section 2 : フロー制御
- Section 3 : APIコンテンツ
- **Section 4 : 並行性**
- Section 5 : オブジェクト指向コンセプト
- Section 6 : コレクション/ジェネリックス
- Section 7 : Java言語の基礎

Section 4: 並行性
スレッドの生成と実行

- スレッドクラスの定義
  a. Threadクラスのサブクラスとして定義
  b. Runnableインタフェースを実装するクラスとして定義



Section 4: 並行性
Threadクラスのサブクラスとして定義

- Threadクラスのサブクラスの定義

```
// Threadクラスを継承して定義
class UseThread extends Thread {
    ...
    // Threadクラスのrun()メソッドをオーバーライド
    public void run() {
        スレッドが実行するコード
        （スレッドに実行させたい処理内容）
    }
}
```

- 作成したスレッドクラスを利用する側

```
class invoker {
    public static void main(String [ ] args) {
        // スレッドオブジェクトの生成
        UseThread  kick  =  new  UseThread();

        // スレッド開始の合図
        kick.start();   //start() から run()メソッドが呼び出される
    }
}
```

Section 4: 並行性
Runnableインタフェースの実装クラスとして定義

- Runnableインタフェースの実装クラスの定義

```
// Runnableインタフェースを実装して定義
class UseRunner implements Runnable {
    ...
    // Runnableインタフェースのrun()メソッドをオーバーライド
    public void run() {
        スレッドが実行するコード（スレッドに実行させたい処理内容）
    }
}
```

- 作成したスレッドクラスを利用する側

```
class invoker {
    public static void main(String [ ] args) {
        // Runnableインタフェース実装クラスのオブジェクト生成
        UseRunner runner = new UseRunner();
        // 生成したスレッドがrunメソッドを参照するときに使用するオブジェクトを生成
        Thread kick = new Thread(runner);
        // スレッド開始の合図
        kick.start();
    }
}
```

問題4:

以下のコードがあります。実行結果はどれですか？

```
11. class Banana implements Runnable {
12.   public void run() {
13.     System.out.println("Banana");
14.   }
15. }
16. public class Test {
17.   public static void main(String [] args) {
18.     Thread t = new Thread(new Banana()) {
19.       public void run() {
20.         System.out.println("Lemon");
21.       }
22.     };
23.     t.start();
24.   }
25. }
```

a. Lemon
b. Banana
c. コードは実行されるが何も出力されない
d. コンパイルエラーになる
e. 実行時に例外が発生する

解答4:

以下のコードがあります。実行結果はどれですか？

```
11. class Banana implements Runnable {
12.   public void run() {
13.     System.out.println("Banana");
14.   }
15. }
16. public class Test {
17.   public static void main(String [] args) {
18.     Thread t = new Thread(new Banana()) {
19.       public void run() {
20.         System.out.println("Lemon");
21.       }
22.     };
23.     t.start();
24.   }
25. }
```

a. Lemon
b. Banana
c. コードは実行されるが何も出力されない
d. コンパイルエラーになる
e. 実行時に例外が発生する

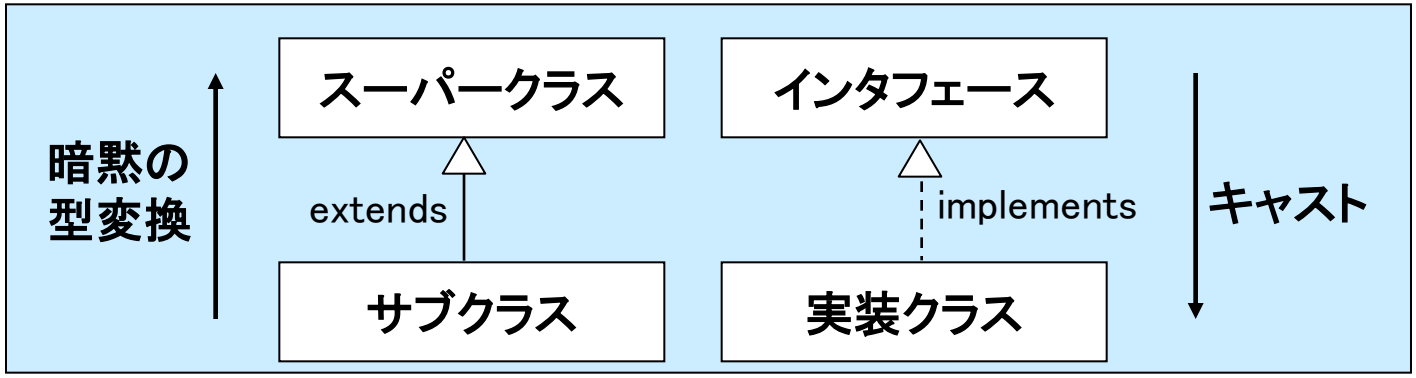
OCJ-P SE 6出題範囲

- Section 1 : 宣言、初期化、スコープ
- Section 2 : フロー制御
- Section 3 : APIコンテンツ
- Section 4 : 並行性
- **Section 5 : オブジェクト指向コンセプト**
- Section 6 : コレクション/ジェネリックス
- Section 7 : Java言語の基礎

Section 5: オブジェクト指向コンセプト
参照型の型変換

- 参照型における型変換ルール1（暗黙の型変換）
  - スーパークラス型 ← サブクラス型のオブジェクト
  - インタフェース型 ← 実装クラス型のオブジェクト
- 参照型における変換ルール2（キャスト）
  - サブクラス型 ← (サブクラス型)スーパークラス型のオブジェクト
  - 実装クラス型 ← (実装クラス型)インタフェース型のオブジェクト

ただしキャストできるのは、あるクラスを暗黙の型変換でスーパークラス型などへ変換したものをもとの型に戻す場合にのみ有効



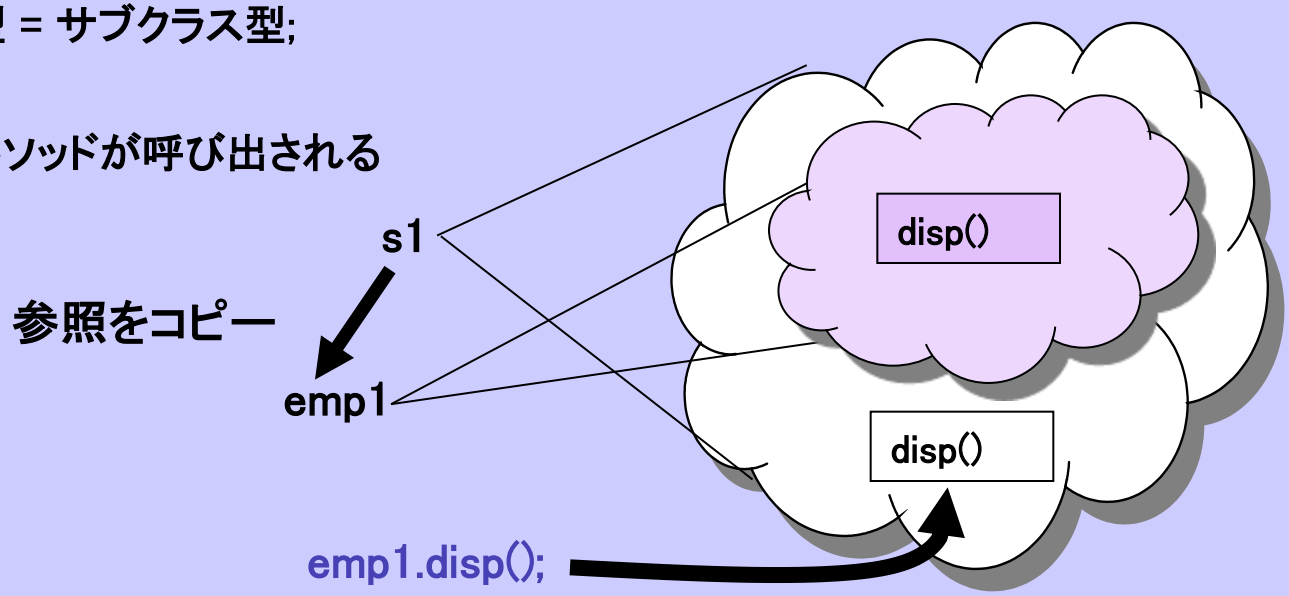
Section 5: オブジェクト指向コンセプト
参照型の型変換(暗黙の型変換)

```
class  Employee { void disp() {…} }
class  Sales  extends  Employee { void disp(){…} }

Employee emp1;
Sales  s1 =  new Sales( );

//スーパークラス型 = サブクラス型;
emp1 = s1;

// オーバーライドメソッドが呼び出される
emp1.disp();
```



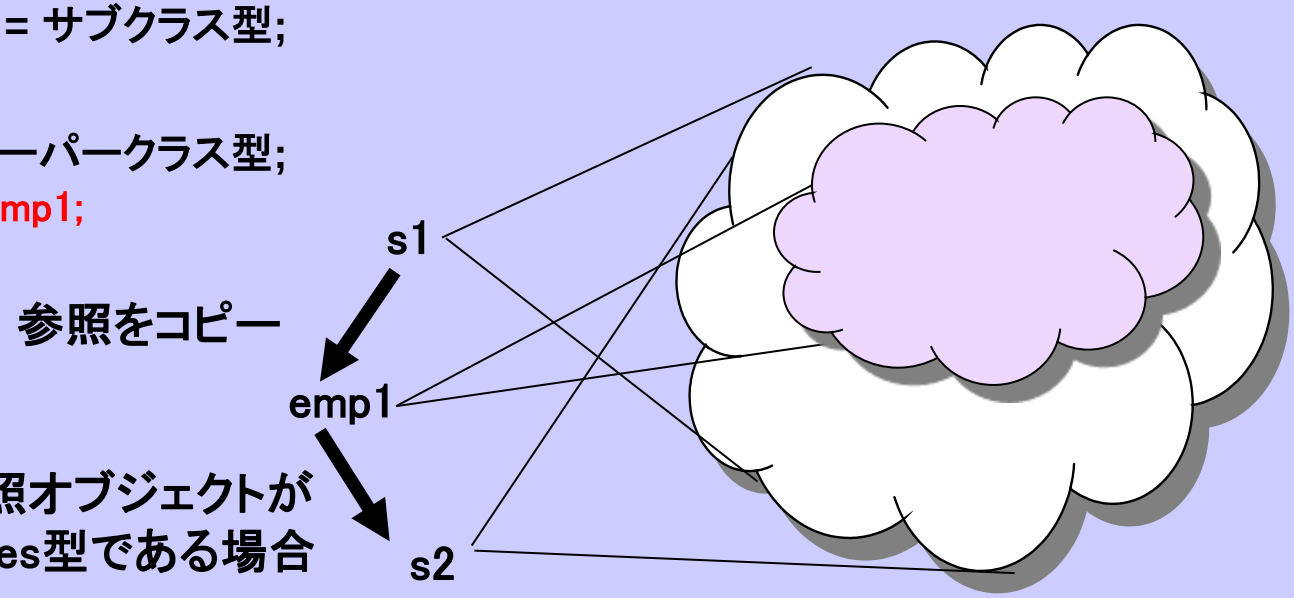
Section 5: オブジェクト指向コンセプト
参照型の型変換(キャスト)

```
class  Employee { … }
class  Sales  extends  Employee { … }

Employee emp1;
Sales  s1 =  new Sales( );

//スーパークラス型 = サブクラス型;
emp1 = s1;

//サブクラス型 = スーパークラス型;
Sales s2 = (Sales)emp1;
```



問題5:

以下のコードがあります。
20 行目に挿入するコードとしてふさわしいものはどれですか？(2 つ選択してください。)

```
11. interface Flyer { void fly(); }
12. class Airplane implements Flyer {
13.   public void fly() { /* some code here */ }
14. }
15. class Helicopter implements Flyer {
16.   public void fly() { /* some code here */ }
17. }
18. public class Test {
19.   public static void main(String [] args) {
20.     // insert code here
21.   }
22. }
```

a. `Airplane ap = new Airplane();ap.fly();`
b. `Airplane a = new Helicopter(); a.fly();`
c. `Flyer f = new Helicopter(); f.fly();`
d. `Flyer f = new Flyer(); f.fly();`
e. `Helicopter h = new Airplane(); h .fly();`

問題5:

以下のコードがあります。
20 行目に挿入するコードとしてふさわしいものはどれですか？(2 つ選択してください。)

```
11. interface Flyer { void fly(); }
12. class Airplane implements Flyer {
13.   public void fly() { /* some code here */ }
14. }
15. class Helicopter implements Flyer {
16.   public void fly() { /* some code here */ }
17. }
18. public class Test {
19.   public static void main(String [] args) {
20.     // insert code here
21.   }
22. }
```

a. Airplane ap = new Airplane();ap.fly();
b. Airplane a = new Helicopter(); a.fly();
c. Flyer f = new Helicopter(); f.fly();
d. Flyer f = new Flyer(); f.fly();
e. Helicopter h = new Airplane(); h .fly();

OCJ-P SE 6出題範囲

- Section 1 : 宣言、初期化、スコープ
- Section 2 : フロー制御
- Section 3 : APIコンテンツ
- Section 4 : 並行性
- Section 5 : オブジェクト指向コンセプト
- **Section 6 : コレクション/ジェネリックス**
- Section 7 : Java言語の基礎

Section 6: コレクション/ジェネリックス
コレクションの種類

- コレクションの種類と特徴

	リスト(List)	セット(Set)	マップ(Map)
構成	Collection I/Fを 継承、実装	Collection I/Fを 継承、実装	Collection I/Fの 継承、実装なし
重複オブジェクト	○	X	キー：X 値　：○
順序付け	○ 添え字をつけて 管理	X 順不同で管理	クラスにより異なる。 キーとリンクして管理。
代表的なクラス	ArrayList LinkedList	HashSet TreeSet	HashMap

Section 6: コレクション/ジェネリックス
ジェネリックス対応クラスと使用方法

- クラスやメソッドを汎用的に使用可能にするための機能

定義方法:
　コレクション名<型パラメータ, …>
使用方法:
　コレクション名<格納する型, …> 変数名 = new コレクション名<格納する型, …>();

例:
```
// ArrayListクラス（抜粋）
class ArrayList<E> extends AbstractList<E>, implements List<E>,… {
    public boolean add(E e) { … }
    …
}
```

使用する側:
```
class Test {
    public static void main(String[] args) {
        ArrayList<String>  list = new ArrayLIst<String>();
        list.add("aaa");    //String型を扱うadd()メソッドとして動作
    }
}
```

問題6:

【挿入コード】の中で、以下のコードの5 行目に挿入して正常にコンパイルされるものはどれですか？

```
 1. import java.util.*;
 2. public class Gen3 {
 3.   public static void go(Set<Dog> d) { }
 4.   public static void main(String [] args) {
 5.     // insert code here
 6.     go(t);
 7.   }
 8. }
 9. class Animal { }
10. class Dog extends Animal { }
```

【挿入コード】

```
s1.    TreeSet t = new TreeSet();
s2.    TreeSet<Dog> t = new TreeSet<Dog>();
s3.    TreeSet<Animal> t = new TreeSet<Dog>();
s4.    TreeSet<Animal> t = new TreeSet<Animal>();
```

a. s1 のみ
b. s2 のみ
c. s1 と s2
d. s1 と s3
e. s1 と s2 と s3
f. s1 と s2 と s4
g. どのコードも正常にコンパイルされる

解答6:

【挿入コード】の中で、以下のコードの5 行目に挿入して正常にコンパイルされるものはどれですか？

```
 1. import java.util.*;
 2. public class Gen3 {
 3.   public static void go(Set<Dog> d) { }
 4.   public static void main(String [] args) {
 5.     // insert code here
 6.     go(t);
 7.   }
 8. }
 9. class Animal { }
10. class Dog extends Animal { }
```

【挿入コード】

```
    s1.  TreeSet t = new TreeSet();
    s2.  TreeSet<Dog> t = new TreeSet<Dog>();
    s3.  TreeSet<Animal> t = new TreeSet<Dog>();
    s4.  TreeSet<Animal> t = new TreeSet<Animal>();
```

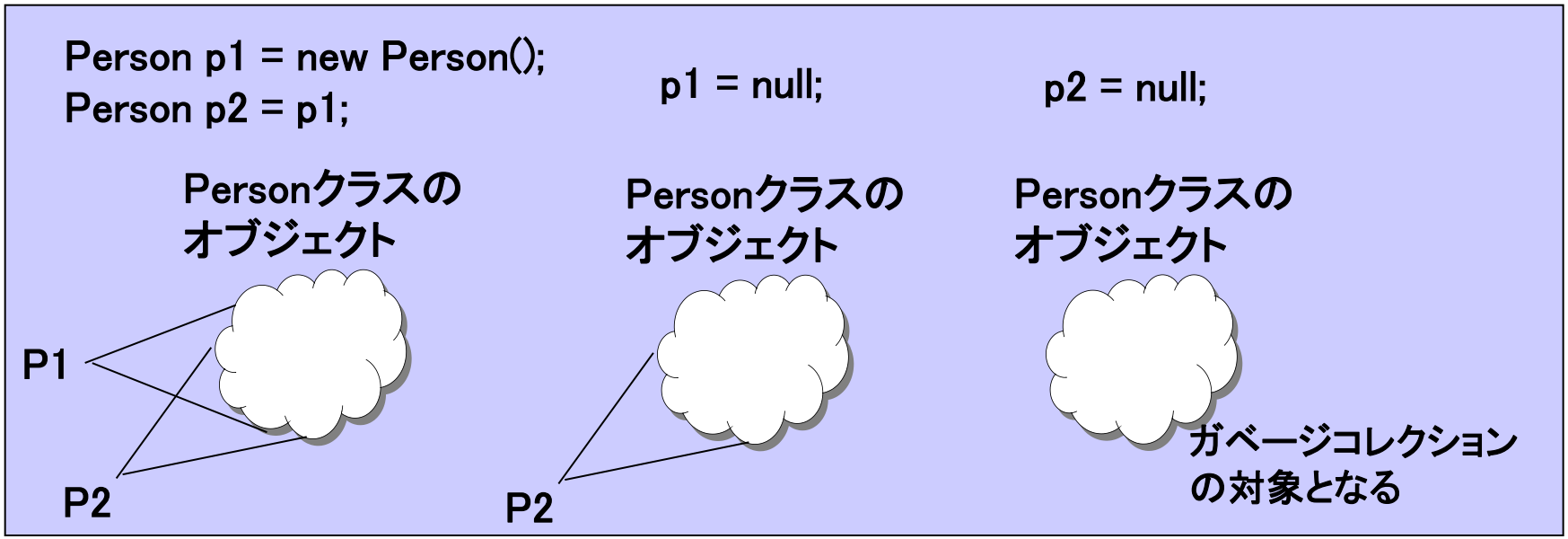
a. s1 のみ
b. s2 のみ
c. s1 と s2
d. s1 と s3
e. s1 と s2 と s3
f. s1 と s2 と s4
g. どのコードも正常にコンパイルされる

OCJ-P SE 6出題範囲

- Section 1 : 宣言、初期化、スコープ
- Section 2 : フロー制御
- Section 3 : APIコンテンツ
- Section 4 : 並行性
- Section 5 : オブジェクト指向コンセプト
- Section 6 : コレクション/ジェネリックス
- **Section 7 : Java言語の基礎**

Section 7: Java言語の基礎
ガベージコレクション

- ガベージコレクションの候補になるタイミングは？
  - 参照変数にnullを代入。対象オブジェクトへの参照がすべてなくなったとき。
- System.gc()メソッド



問題7:

以下のコードがあります。
12 行目まで処理された後、ガベージコレクションの対象となるオブジェクトはいくつですか？

```
 1. class Rubbish {
 2.   public static void main(String [] args) {
 3.     Rubbish r1 = new Rubbish();
 4.     Rubbish r2 = new Rubbish();
 5.     Rubbish r3 = new Rubbish();
 6.     Rubbish r4 = r2;
 7.     Rubbish r5 = r4;
 8.     r2 = null;
 9.     r4 = r2;
10.     r1 = r5;
11.     // do stuff
12.   }
13. }
```

a. 0
b. 1
c. 2
d. 3
e. 4

解答7:

以下のコードがあります。
12 行目まで処理された後、ガベージコレクションの対象となるオブジェクトはいくつですか？

```
 1. class Rubbish {
 2.   public static void main(String [] args) {
 3.     Rubbish r1 = new Rubbish();
 4.     Rubbish r2 = new Rubbish();
 5.     Rubbish r3 = new Rubbish();
 6.     Rubbish r4 = r2;
 7.     Rubbish r5 = r4;
 8.     r2 = null;
 9.     r4 = r2;
10.     r1 = r5;
11.     // do stuff
12.   }
13. }
```

a. 0
b. 1
c. 2
d. 3
e. 4

ePractice（オンライン問題集）

Java、Oracle Solaris資格試験の受験を目指す方におすすめの公式eラーニング・サービス登場！

http://education.oracle.co.jp/ePractice/

- Java、Oracle Solaris認定資格試験に準じた問題をオンラインで学習できる、新しいeラーニング・サービス
- ASP型のサービスなのでインターネットとブラウザがあれば、どこからでもアクセス可能
- 個々の学習スタイルで、選べる2つの学習モード

問題集モード

試験範囲のカテゴリー別に、全ての問題を学習することができます。正解および解説を確認しながら学習を進めることができるため、カテゴリー毎の学習・理解度の確認に最適です。

模擬試験モード

豊富な問題数よりランダムに出題される模擬試験です。テストを終えるとその場で採点が行なわれ、正誤判定及び解答、詳細解説が表示されます。

問題が毎回変わるため、反復しておこなうことが可能。また試験前の確認チェックはもちろん、学習前のスキル診断としても有効です。

ePractice（オンライン問題集）一覧

製品名	問題総数	学習期間	価格（税込）
ePractice(オンライン問題集) : Oracle認定Javaアソシエイツ	約150問	180日間	7,418円
ePractice(オンライン問題集) : Oracle認定JavaプログラマSE 6	約120問	180日間	7,418円
ePractice(オンライン問題集) : Oracle認定Webコンポーネント ディベロッパEE 5	約120問	180日間	7,418円
ePractice(オンライン問題集) : Oracle認定Oracle Solaris 10 システム管理者, Part I	約180問	180日間	7,418円
ePractice(オンライン問題集) : Oracle認定Oracle Solaris 10 システム管理者, Part II	約180問	180日間	7,418円

学習内容（例）

ePractice(オンライン問題集) : Oracle認定Javaアソシエイツ	
【問題集】1: オブジェクト指向の基礎	【模擬試験】テスト1
【問題集】2: UML 表記	【模擬試験】テスト2
【問題集】3: Java 言語の基礎	【模擬試験】テスト3
【問題集】4: Java 言語のアルゴリズム	【模擬試験】実力診断テスト
【問題集】5: Java言語の基本コマンドとJavaパッケージ	操作ガイド
【問題集】6: Javaプラットフォームと関連テクノロジー	
【問題集】7: クライアントサイド・テクノロジー	
【問題集】8: サーバーサイド・テクノロジー	

認定資格バリューパッケージ

おトクに学んで資格を取得。安心・充実のパッケージ登場

http://education.oracle.co.jp/ValuePackage/

- Java、Oracle Solaris資格取得に効果的な学習コンテンツのフルパッケージ
- 個別に購入するよりも5% OFF！
- 受験チケットにはもれなく再試験特典つき

推奨研修コース

・各認定資格に対応した推奨研修コースです。経験豊富なインストラクターが分かりやすく講義をすすめ、受講者の理解度を高めます。また知識の定着に効果的な実機付きの演習もあり、資格取得の枠にとらわれない、スキル向上をお約束します。

ePractice（オンライン問題集）

・各認定資格に対応した試験の擬似問題がとけるオンライン問題集です。180日間、何度でもアクセスして問題にトライすることが可能です。実際の試験に慣れるためにも最適な自習コンテンツです。

再試験特典付き受験チケット

・試験に合格できなかった場合でも、再度同じ試験にトライできる再試験特典がついています。

認定資格バリューパッケージ一覧

バリューパッケージ	税込価格（5%　割引）	コンテンツ
Oracle認定Javaアソシエイツ バリューパッケージ	187,695円	Java テクノロジー/プログラミング基礎　(SL-019)
		ePractice（オンライン問題集）: Oracle認定Javaアソシエイツ
		オラクル認定試験 受験チケット（再試験特典付き）
Oracle認定Javaプログラマ バリューパッケージ	286,447円	Javaプログラミング I　(SL-275-1-V2)
		Javaプログラミング II　(SL-275-2-V2)
		ePractice（オンライン問題集）: Oracle認定JavaプログラマSE 6
		オラクル認定試験 受験チケット（再試験特典付き）
Oracle認定Webコンポーネント ディベロッパ バリューパッケージ	237,071円	サーブレット/JSP を使用した Web Component 開発 (SL-314-V2)
		ePractice（オンライン問題集）: Oracle認定WebコンポーネントディベロッパEE 5
		オラクル認定試験 受験チケット（再試験特典付き）
Oracle認定Oracle Solaris10 システム管理者, Part I バリューパッケージ	368,741円	Solaris 10 システム管理 I (SA-200-S10-1)
		Solaris 10 システム管理 II (SA-200-S10-2)
		ePractice(オンライン問題集) : Oracle認定Oracle Solaris10 システム管理者, Part I
		オラクル認定試験 受験チケット（再試験特典付き）
Oracle認定Oracle Solaris10 システム管理者, Part II バリューパッケージ	368,741円	Solaris 10 システム管理 III (SA-202-S10-1)
		Solaris10 システム管理 IV (SA-202-S10-2)
		ePractice(オンライン問題集) : Oracle認定Oracle Solaris10 システム管理者, Part II
		オラクル認定試験 受験チケット（再試験特典付き）

Monthly メールマガジン:
Oracle Unversity eNewsLetter

オラクルの資格とトレーニングの最新情報をお届けします

http://education.oracle.co.jp/eNewsLetter/

注目の研修コースや、ORACLE MASTER
をはじめとしたオラクル認定資格など、
スキルアップに役立つ情報を豊富に掲載。
最新情報が一挙に入手できます。

さらに、スキルアップに役立つお得なキャンペーン
情報なども、いち早くご提供します。

メルマガ コンテンツ

- スキルアップに役立つイベント開催情報
- お得なキャンペーン情報
- オラクル製品の最新情報や対応研修コースのご紹介
- ORACLE MASTERやその他認定資格の最新情報
- 解説付き模擬試験「Try! オラクル認定資格」

Copyright© 2010, Oracle. All rights reserved.

OTN×ダイセミ　でスキルアップ!!

・一般的な技術問題解決方法などを知りたい！
・セミナ資料など技術コンテンツがほしい！

Oracle Technology Network(OTN)を御活用下さい。

http://otn.oracle.co.jp/forum/index.jspa?categoryID=2

一般的技術問題解決にはOTN掲示版の「Java」をご活用ください

※OTN掲示版は、基本的にOracleユーザー有志からの回答となるため100%回答があるとは限りません。
ただ、過去の履歴を見ると、質問の大多数に関してなんらかの回答が書き込まれております。

http://www.oracle.com/technology/global/jp/ondemand/otn-seminar/index.html

過去のセミナ資料、動画コンテンツはOTNの「OTNセミナー　オンデマンド コンテンツ」へ

※ダイセミ事務局にダイセミ資料を請求頂いても、お受けできない可能性がございますので予めご了承ください。
ダイセミ資料はOTNコンテンツ　オン　デマンドか、セミナ実施時間内にダウンロード頂くようお願い致します。

OTNセミナー　オンデマンド　コンテンツ

ダイセミで実施された技術コンテンツを動画で配信中!!

ダイセミのライブ感はそのままに、お好きな時間で受講頂けます。

twitter

最新情報つぶやき中

oracletechnetjp

- 人気コンテンツは？
- お勧め情報
- 公開予告　など

OTN　オンデマンド　検索

※掲載のコンテンツ内容は予告なく変更になる可能性があります。
期間限定での配信コンテンツも含まれております。お早めにダウンロード頂くことをお勧めいたします。

Oracle エンジニアのための技術情報サイト

オラクルエンジニア通信

http://blogs.oracle.com/oracle4engineer/

twitter

最新情報つぶやき中

oracletechnetjp

- 技術資料
  - ダイセミの過去資料や製品ホワイトペーパー、スキルアップ資料などを多様な方法で検索できます
  - キーワード検索、レベル別、カテゴリ別、製品・機能別
- コラム
  - オラクル製品に関する技術コラムを毎週お届けします
  - 決してニッチではなく、誰もが明日から使える技術の「あ、そうだったんだ！」をお届けします

オラクルエンジニア通信

こんな資料が人気です

✓ 5ヶ月連続で「RAC/ASMインストール資料」が第一位。
根強い人気のチュートリアル系コンテンツですが、新たに「Oracle Enterprise Managerインストール資料」が第四位にランクインしました。

✓ パフォーマンス・チューニング コンテンツを集めた特集ページも好評です。

ITプロジェクト全般に渡る無償支援サービス

Oracle Direct Conciergeサービス

■パフォーマンス診断サービス
- Webシステム　ボトルネック診断サービス NEW
- データベースパフォーマンス 診断サービス

■移行支援サービス
- SQL Serverからの移行支援サービス
- DB2からの移行支援サービス
- Sybaseからの移行支援サービス
- MySQLからの移行支援サービス
- Postgre SQLからの移行支援サービス
- Accessからの移行支援サービス
- Oracle Application ServerからWeblogicへ移行支援サービス NEW

■システム構成診断サービス
- Oracle Database構成相談サービス
- サーバー統合支援サービス
- 仮想化アセスメントサービス
- メインフレーム資産活用相談サービス
- BI EEアセスメントサービス
- 簡易業務診断サービス

■バージョンアップ支援サービス
- Oracle Databaseバージョンアップ支援サービス
- Weblogic Serverバージョンアップ支援サービス NEW
- Oracle Developer/2000(Froms/Reports) Webアップグレード相談サービス

オラクル社のエンジニアが 直接ご支援します
お気軽にご活用ください!

オラクル　無償支援　検索

ORACLE

あなたにいちばん近いオラクル

Oracle Direct

まずはお問合せください

Oracle Direct　検索

システムの検討・構築から運用まで、ITプロジェクト全般の相談窓口としてご支援いたします。

システム構成やライセンス/購入方法などお気軽にお問い合わせ下さい。

Web問い合わせフォーム

専用お問い合わせフォームにてご相談内容を承ります。
http://www.oracle.co.jp/inq_pl/INQUIRY/quest?rid=28

※フォームの入力には、Oracle Direct Seminar申込時と同じ
　ログインが必要となります。
※こちらから詳細確認のお電話を差し上げる場合がありますので、ご登録されている連絡先が最新のものになっているか、ご確認下さい。

フリーダイヤル

0120－155－096

※月曜~金曜　9:00~12:00、13:00~18:00
（祝日および年末年始除く）

ORACLE