# Architecture Specialization サンプル試験

## Sample Exam

## 開始前の注意事項

このサンプル試験には、OutSystems 11 Architecture Specialization試験の準備に役立つ問題が10問掲載されています。できるだけ本番に近い試験環境を準備することを推奨します。

- 周りに人がいない、静かな部屋を確保します
- このドキュメントの最終ページ以外を印刷します。
- ストップウォッチまたはタイマーを使用して40分（推奨）で行います。

このドキュメントの最終ページに解答が記載されています。解答は先に見ないようにし、試験終了後に、正誤を確認するために使用するようにしてください。

# サンプル試験中の注意事項

本番に近い試験環境で取り組むために、以下の点を心がけてください。

- 各問題と選択肢を慎重に読みましょう。
- あわてずに取り組みましょう。後から問題を見直して解答を変更することもできます。
- 最後に見直したい問題には印を付けましょう。
- 解答は1問につき1つだけ選択してください。正解は1つだけです
- すべての問題に解答しましょう。未解答の問題は得点になりません。
- 試験中はすべての電子機器の電源をオフにしましょう。
- 試験中は他の資料を使用したり参照したりしないでください。

# サンプル試験終了後の注意事項

試験終了後、このドキュメントの最終ページに記載された解答を参照して答え合わせを行い、正解の合計数を数えます。合格点は70%以上なので、11問以上正解する必要があります。間違えた問題に関しては、教材でそのトピックに関する説明を復習することを推奨します。

# サンプル試験問題

**1.** 次のうち、Architecture Canvas導入のメリットとして正しいものはどれですか。

- ☐ **A.** 検証ツールを活用しながら、体系的なアプローチでアーキテクチャ設計を行うことができる。
- ☐ **B.** アーキテクチャ原理の適用や修正を自動的に行うことができる。
- ☐ **C.** コラボレーションを促進しやすくなり、ビジネスユーザーの理解度も高められる。
- ☐ **D.** 検証が不要であり、アーキテクチャ設計を迅速に行うことができる。

---

**2.** OutSystemsで、関連しないコンセプトを同じモジュールにまとめる必要はありますか。

- ☐ **A.** ある。参照の数を減らし、デプロイを簡素化できる。
- ☐ **B.** ない。コンセプトのライフサイクルの独立性がなくなり、コンシューマに余計な影響が生じる。
- ☐ **C.** ある。循環参照を避け、生成されるコードのフットプリントを減らすことができる。
- ☐ **D.** ない。両方のコンセプトが必要になり、コンシューマが無駄に複雑化する。

---

**3.** 「OutSystems バージョン11では画面への参照は弱い依存関係となるため、Architecture Canvasの検証ルールに違反しないよう、ビジネス関連の画面は基盤レイヤーモジュールで使用することを推奨する」という文について考えます。 次のうち、説明として正しいものはどれですか。

---

- ☐ **A.** この文はすべてのビジネス関連の画面について正しい。画面への参照はすべて弱い依存関係であるため、上方参照を引き起こさないよう基盤レイヤーモジュール内で画面を使用する必要がある。
- ☐ **B.** この文は正しくない。画面への参照はすべて弱い依存関係であるが、基盤レイヤーモジュールで使用する必要があるのはビジネス関連以外の画面のみである。
- ☐ **C.** この文はビジネス関連のWeb画面については正しい。モバイル画面への参照は引き続き強い依存関係とみなされるため、基盤レイヤーモジュールで使用してはならない。
- ☐ **D.** この文は正しくない。画面への参照はすべて弱い依存関係であるが、ビジネス関連の画面は基盤レイヤーモジュールで使用してはならない。

---

**4.** エンドユーザーモジュール間で弱いサイドリファレンスがあるとします。この参照を解消し、アーキテクチャ違反を回避する方法として最適なものはどれですか。

- ☐ **A.** 参照先を外部URLに置き換え、REST APIを介してアクションを利用する。
- ☐ **B.** エンドユーザーモジュール間の弱いサイドリファレンスは許容されるため、この参照を解消する必要はない。
- ☐ **C.** 参照を避けるため、常に利用対象の要素をプロデューサからコンシューマに移動する。
- ☐ **D.** 利用対象の要素を特定し、参照先を除外して、再利用可能なコア/基盤モジュールに移動する。

---

**5.** 次のうち、計算エンジン（_Eng）モジュールを作成する理由として妥当なものはどれですか。

- ☐ **A.** 複数のBL内に実装されている複雑な各種ロジックを抽象化するため。
- ☐ **B.** エンティティやCRUD編集を伴うコンセプトをサポートするため。
- ☐ **C.** 複雑な計算（保険シミュレータなど）をサポートするため。
- ☐ **D.** 再利用可能なコアウィジェットモジュールアクションをサポートするため。

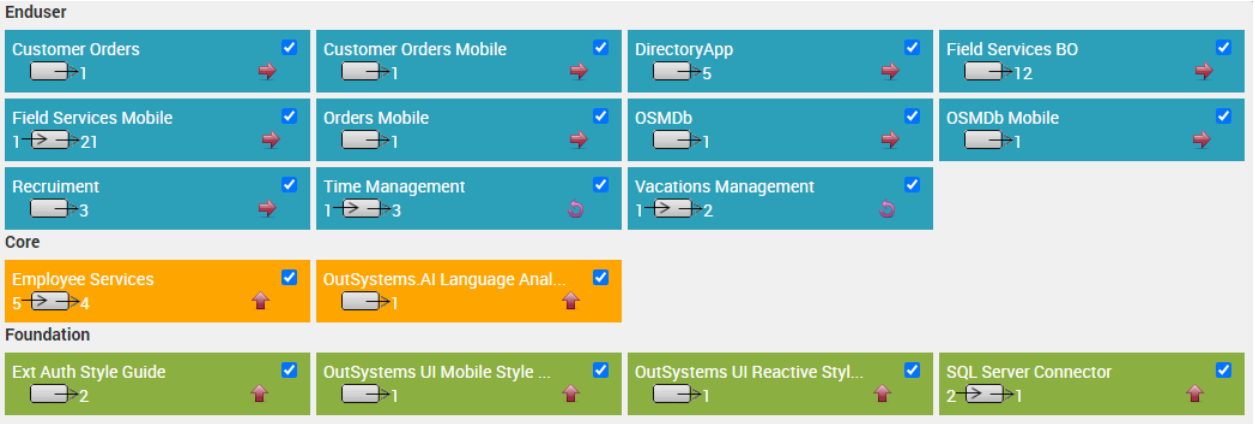
---

**6.** アプリケーションで命名規則を採用することが重要なのはなぜですか。

- ☐ **A.** 各モジュールの性質を明確にできるため。
- ☐ **B.** 参照アーキテクチャを適用できるため。
- ☐ **C.** パターンを正規化できるため。
- ☐ **D.** いずれの選択肢も正しい。

---

7. 次のようなDiscoveryのスクリーンショットと検出項目について考えます。次のような検出項目がある場合、最初に解決すべきものはどれですか。




- ☐ A. コアサービスモジュールおよびエンドユーザーモジュール間での上方参照。
- ☐ B. 基盤モジュールおよびコアサービスモジュール間での上方参照。
- ☐ C. エンドユーザーモジュール間での参照。
- ☐ D. アーキテクチャの検出項目を解決する際、順序は気にしなくてよい。

8. モジュール間で要素を移動する際、特に注意と慎重な作業を要するのはどの要素ですか。

- ☐ A. アクション、ストラクチャ、ブロックなどデータの永続性がない要素。これらは永続性がないため、プロセス中に意図せず削除されるおそれがある。
- ☐ B. エンティティなどデータの永続性がある要素。エンティティの論理的な定義のみが転送され、物理的には別のデータベーステーブルであるため。
- ☐ C. 特にない。モジュール間で移動されるすべての要素の整合性が維持される。
- ☐ D. タイマー、サイトプロパティ、ロールなど構成可能な要素のみ。データの永続性があり、開発環境と本番環境で異なる要素はこれらのみであるため。

9. アーキテクチャ設計をサポートするためのマルチレイヤーフレームワークは、何を実現するうえで欠かせませんか。

- ☐ A. 再利用可能なサービスの適切な抽象化の促進。
- ☐ B. 要素間のライフサイクルの独立性の最適化。
- ☐ C. 変更の影響の最小化。
- ☐ D. いずれの選択肢も正しい。

**10.** 次のシナリオのうち、カスタムスタイルガイドを実装する必要があるものはどれですか。

- ☐ **A.** 既存のテーマをアプリケーションで利用できないすべてのシナリオ。元のテンプレートのブロックやアクションを活用して開発を高速化するため、利用可能なもののなかで最も複雑なテンプレートを複製してスタイルガイドを実装する必要がある。
- ☐ **B.** 既存のテーマをアプリケーションで利用できないすべてのシナリオ。必要最低限のストラクチャが含まれる、できる限り基本的なテンプレートから開始してスタイルガイドを実装する必要がある。
- ☐ **C.** アプリケーションのデプロイの独立性を確保し、他のアプリケーションとの相互依存を低水準に保つ必要があるすべてのシナリオ。
- ☐ **D.** すべてのシナリオ。すべてのプロジェクト/アプリケーションに独自のカスタムスタイルガイドが必要である。

# 解答

**1. A**

**2. B**

**3. D**

**4. B**

**5. C**

**6. D**

**7. C**

**8. B**

**9. D**

**10. B**