9. XP(eXtreme Programming)

〈XP 정의〉

짧은 주기의 반복으로 고객의 요구 **변화**에 **신속**하게 **대응**하여 위험을 줄이고
고객 관점의 고품질 소프트웨어를 신속하게 전달하는 **애자일**(agile) 방법론 기법이다.

◈ XP 핵심가치(core value)

단순성 **(simplicity)**	• 간단한 설계 부분은 바로 작성하고 • 복잡한 설계 부분은 지속적인 **리팩토링으로 복잡성을 줄이도록 노력**한다. • 부가적 기능, 사용되지 않는 구조와 알고리즘을 배제한다.
피드백 **(feedback)**	• 빠른 피드백을 기본 원칙으로 하며, **해결할 수 있는 일을 먼저 처리**한다. • 지속적인 테스트와 반복적으로 결함을 수정하고, 빠른 피드백을 실시한다. • 피드백은 의사소통의 핵심이며 단순성에도 기여한다.
의사소통 **(communication)**	• 의사소통은 팀 개발에 있어서 가장 중요한 요소이다. • 의사소통은 한 팀이라는 느낌을 만들고 효과적 협동을 위한 중요 부분이다. • **개발자, 관리자, 고객** 간의 원활한 의사소통이다.
용기 **(courage)**	• 실패하는 해결책은 버리고 **새로운 해결책을 찾아 나서는 용기**이다. • 용기는 단순함에 도움을 주고 • 용기는 구체적이고 진실한 답변을 추구하는 피드백을 낳게 한다. • 용기는 고객의 요구사항 변화에 능동적인 대처를 낳게 한다.
존중 **(respect)**	• 프로젝트 관리자는 모든 **팀원의 기여를 존중**해야 한다. • 개발자의 역량을 존중하고 충분한 권한과 권리 부여한다.

◈ 사용자 스토리(user story)

- 사용자 스토리는 **사용자 요구사항을 의미**한다.
- 사용자 스토리는 요구사항 수집, 의사소통 도구가 된다.**(UML의 유스케이스와 같은 목적)**
- 사용자 스토리는 릴리즈 계획을 작성하기 위한 단위가 된다.
- 사용자 스토리는 기능 단위로 필요한 내용을 간단하게 기술한다.
- 필요시, 사용자 스토리에는 간단한 **테스트 사항**도 기술할 수 있다.
- 사용자 스토리 : 사용자가 시스템에서 무엇을 필요로 하는지를 기술한다.**(인수테스트에 사용)**

◈ 구조적 스파이크(architectural spike)

- 어려운 요구사항 혹은 잠재적인 **해결책(solution)을 위해 작성**하는 간단한 프로그램
- 사용자 스토리의 신뢰성을 증대시키고, 기술적인 문제에 대한 위험을 줄이기 위한 목적이다.

◈ XP의 실무 관행(12가지)

- **관리(3)** : 계획 게임, 작은 릴리즈, 메타포
- **구현(3)** : 단순한 디자인, 테스트 주도 개발, 리팩토링
- **개발(4)** : 짝 프로그래밍, 코딩표준, 공동 소유권, 계속적인 통합
- **환경(2)** : 주 40시간 근무, 현장의 고객

↓

계획 게임 **(planning game)**	• planning game = planning process • XP는 보통 2주 단위로 계획을 세우는데, 중요한 **2가지**가 있다. • **첫 번째**, 이번 반복(iteration)에서 어떤 개발 과정을 끝마칠 것인가? • **두 번째**, 그 이후 개발 반복에서는 무엇을 할 것인가?
작은 릴리스 **(small release)**	• 필요한 기능만 갖춘 간단한 시스템을 빠르게 만든다. • 2주정도의 아주 **짧은 주기**로 자주 새로운 버전을 **배포**한다.
메타포 **(metaphor)**	• 메타포는 공통의 name system이다. • 메타포는 개발 및 **의사소통** 과정에서 공통된 개념을 공유하게 함
단순한 디자인 **(simple design)**	• 현재의 요구사항을 만족하도록 가능한 단순하게 설계 • 현재 당장 필요하지 않은 미래를 고려한 설계는 최대한 배제한다.
테스팅 **(testing)**	• **테스트 주도 개발**(Test-driven development, TDD)을 한다. • 각 사용자 스토리에 대해 테스트 케이스를 작성한다. • 개발자가 먼저 단위 테스트를 실시하고, • 고객은 요구사항 반영 여부를 확인한다.
리팩토링 **(refactoring)**	• 지속적으로, 최대한 많은 코드를 리팩토링 한다. • 리팩토링은 프로그램의 기능을 바꾸지 않는다. • 리팩토링은 단순화, 유연성, 중복제거, 의사소통을 향상시킨다.
짝 프로그래밍 **(pair programming)**	• 하나의 컴퓨터에 **두 사람**이 앉아서 같이 프로그램 한다. • 2명 혹은 그 이상의 프로그래머가 함께 코딩한다.
코딩표준 **(coding standard)**	• 코드는 표준화된 관례에 따라 작성되어야 한다. • 표준화된 코드는 팀원들 사이의 의사소통을 향상시킨다.
공동 소유권 **(collective ownership)**	• 모든 개발자들이 **전체 코드에 대한 공동 책임**을 가진다. • 개발자 누구든지 코드 변경이 가능하다.
계속적인 통합 **(continuous integration)**	• 작업이 완료되자마자 전체 시스템에 통합되도록 한다. • 하루에 몇 번이라도 시스템을 통합할 수 있다.
주 40시간 근무 **(40-hour week)**	• 일주일에 **40시간** 이상은 일하지 않도록 규칙으로 정한다. • 그리고, 2주 연속으로 오버타임 하지 않도록 한다. • 초과 근무는 낮은 품질, 보통의 생산성만 양성할 뿐이다.
현장의 고객 **(on-site customer)**	• 개발자의 질문에 즉시 대답해 줄 수 있는 **고객**이 필요하다. • 고객을 프로젝트에 풀타임으로 상주시킨다. • 고객도 개발팀의 일원이다.

① 게임 계획(planning game) - game은 일, 사업이라는 뜻도 있음

- planning game = planning process
- XP는 보통 2주 단위로 계획을 세우는데(프로토타입 작성), 중요한 2가지가 있다.
  → **첫 번째**, 이번 반복(iteration)에서는 어떤 개발 과정을 끝마칠 것인가?
  → **두 번째**, 그 이후 개발 반복에서는 무엇을 할 것인가?
- 작성된 **프로토타입**을 가지고 고객과 함께 검토 혹은 회의를 실시한다.
  → 무엇이 얼마만큼 개발이 되었는지 혹은 알맞은 방향으로 개발이 진행되는지 검사
- 사용자 스토리를 이용해서 다음 릴리스(release)의 범위를 빠르게 결정한다,
- XP에서는 반복적인 **점증적 계획**(incremental planning)을 수립한다.
- XP에서는 소작업으로 분할한다.
- 소작업을 이용하여 스케줄링, 비용 산정 등을 실시한다.

② 작은 릴리스(small release)

- 영어 release는 (대중들에게) 공개[발표]하는 것이다. release는 고객에게 **배포**하는 것이다.
- 개발자는 고객에게 주기적으로 프로토타입을 보여준다.
- 고객은 제한된 기능을 가지고 있지만 실제로 작동이 되는 데모 모델을 볼 수가 있다.
- 고객은 추가 사항을 요구할 수도 있다.
- 개발자는 현재까지 개발 상황이 올바른 길로 가고 있음을 알 수 있다.

③ 메타포(metaphor)

- metaphor의 사전적인 의미는 은유(隱喩, metaphor) 또는 암유(暗喩)이다.
- 은유는 사물의 본뜻을 숨기고 주로 보조관념들만을 간단하게 제시한다.
- 메타포는 모든 사람들이 시스템의 작동 방법에 대해 이야기할 수 있는 스토리이다.
  → 모든 사람들은 **고객, 개발자, 관리자** 등을 의미한다.
- 메타포는 문장 형태로 시스템 아키텍처를 기술한다.
- 메타포는 고객과 개발자 사이의 **의사소통 언어**이다.
- 메타포는 공통 이름 체계의 시스템 서술서를 가짐으로 개발과 의사소통을 돕는다.
- 메타포는 전체 시스템이 어떻게 진행되는지를 쉽게 표현하는 것이다.
- 메타포는 고객을 포함한 모든 개발자를 위해 쉬운 스토리로 작성된다.
- 메타포는 최종적으로 개발 되어야 할 시스템의 구조를 조망하는 것이다.(조감도)

④ 단순한 설계(simple design)

- XP에서 모든 코딩은 가능한 단순하게 할 것을 강조한다.
- XP는 미래를 고려한 설계는 최대한 배제한다.
- 복잡하면, 버그가 발생하면 쉽게 처리를 하지 못하고 오랜 시간이 걸리게 된다.

⑤ 테스트 주도 개발(Test-driven development, TDD)

- TDD는 개발하기 전에 **테스트 케이스를 먼저 작성**하는 개발하는 방법론을 말한다.
- TDD는 코드를 작성하기 전에 테스트 케이스를 먼저 작성한다.
- TDD는 먼저 작성된 테스트 케이스에 맞추어 실제 개발하는 방법론을 말한다.
  → 개발 후에 계획대로 완성되었는지 테스트 케이스를 작성하여 테스트하는 방식이 아니다.
- TDD는 각 사용자 스토리에 대해 테스트 케이스를 작성한다.
- TDD는 초기적 결함을 점검하는 자동화된 테스트 케이스를 작성한다.
- TDD는 잠재된 상황은 무시하고 테스트 케이스만을 완벽하게 수행하는 것을 목표로 한다.
  → 해서, 주어진 목표를 매우 빠르게 완료할 수 있다.
- TDD는 자체적으로 하나의 테스트가 완전하지 않다는 것을 가정하고 있다.
  → 1차 테스트를 완료되면, 다음에는 새로운 확장된 테스트 케이스를 작성한다.
  → 확장된 테스트 케이스를 통과하기 위한 개발 과정을 끊임없이 반복한다.
  → 최종적으로는 큰 규모의 프로젝트를 완성해가는 것이다.

〈TDD 장점〉

- 프로그래밍 시간이 단축된다.
  → 테스트 케이스 작성 시간이 포함되지만 전체 작업 시간은 줄어든다.
  → 이유는, 프로그래밍에서 대부분 시간이 디버깅에 투입되는데,
  TDD는 디버깅 범위를 단위 안으로 제한함으로 디버깅 노고를 크게 줄여준다.
- 코드의 유지보수가 용이해진다
  → 이유는, 테스트 케이스가 존재하기 때문이다.
  → 테스트하기 쉬운 코드는 품질이 높아지고, 다시 읽기도 편하다.
- 새로운 코드나 표준 라이브러리의 컴포넌트를 사용하는 개발에서 가치가 있다.

⑥ 리팩토링(refactoring)

- 지속적으로, 최대한 많은 코드를 리팩토링 한다.
- 리팩토링은 프로그램의 기능을 바꾸지 않는다.
- 리팩토링은 단순화, 유연성, 중복제거, 의사소통을 향상시킨다.

⑦ 짝 프로그래밍(pair programming)

- 2명 혹은 그 이상의 프로그래머가 함께 코딩한다.
- 하나의 컴퓨터에 두 사람이 앉아서 같이 프로그램 한다.
- 2명의 프로그래머가 함께 코딩과 테스트를 하면서 개발할 수도 있고,
- 한명은 코딩을 하고, 한명은 테스트에만 집중할 수도 있다.

• 짝 프로그래밍은 코딩과 **코드 리뷰**를 동시에 적용하는 것과 같다.
• 짝 프로그래밍은 화면・키보드・마우스를 공유하므로 코딩 과정 자체가 리뷰 과정이다.

〈코드 리뷰(code review, 코드 검토)〉
・코드 리뷰는 특정 개발자가 작성한 코드를 다른 개발자가 정해진 방법으로 검토하는 것
・코드 리뷰는 동료가 평가하는 것이다.
・소프트웨어 개발 과정에서 간과됐던 오류를 검출/수정하도록 원시코드를 체계적으로 검사

⑧ 코딩표준(coding standard)
• 코드는 표준화된 관례에 따라 작성되어야 한다.
• 표준화된 코드는 팀원들 사이의 의사소통을 향상시킨다.

⑨ 공동 소유권(collective ownership)
・모든 개발자들이 전체 코드에 대한 공동 책임을 가진다.
・개발자 누구든지 어떤 코드라도 변경할 수 있다.

⑩ 계속적 통합(continuous integration)
• 작업이 완료되자마자 전체 시스템에 통합되도록 한다.
• 하루에 몇 번이라도 시스템을 통합할 수 있다.

⑪ 주 40시간 작업(40-hour week)
• 일주일에 40시간 이상은 일하지 않도록 규칙으로 정한다.
• 그리고, 2주 연속으로 오버타임 하지 않도록 한다.
• 초과 근무는 낮은 품질, 보통의 생산성만 양성할 뿐이다.

⑫ 현장의 고객(on-site customer)
• 개발자의 질문에 즉시 대답해 줄 수 있는 고객이 필요하다.
• 고객을 프로젝트에 풀타임으로 상주시킨다.
• 고객도 개발팀의 일원이다.
• 시스템 요구사항을 전달할 책임이 고객에게 존재해야 한다.
• XP에서는 반복적으로 고객 테스트를 거친다.
• 이를 통해서 개발자가 잘못 이해하고 있었던 부분에 대해서 수정을 거친다.
• 최종적으로 고객의 요구에 부합하는 소프트웨어를 만들어 낸다.

기출문제 분석

1. 다음 특징을 가지는 소프트웨어 개발 방법론은? [2021년 국가 7급]

○ 리팩토링(refactoring)　　○ 사용자 스토리(user story)
○ 소규모 릴리즈(small release)　　○ 짝 프로그래밍(pair programming)
○ 테스트 우선 개발(test-first development)

---

① COCOMO　　② 익스트림 프로그래밍(XP)
③ 통합 프로세스(UP)　　④ 폭포수(waterfall) 모델

♤ 익스트림 프로그래밍(XP)

계획 게임 (planning game)	• planning game = planning process • XP는 보통 2주 단위로 계획을 세우는데, 중요한 **2가지**가 있다. • **첫 번째**, 이번 반복(iteration)에서 어떤 개발 과정을 끝마칠 것인가? • **두 번째**, 그 이후 개발 반복에서는 무엇을 할 것인가?
작은 릴리스 (small release)	• 필요한 기능만 갖춘 간단한 시스템을 빠르게 만든다. • 2주정도의 아주 **짧은 주기**로 자주 새로운 버전을 **배포**한다.
메타포 (metaphor)	• 메타포는 공통의 name system이다. • 메타포는 개발 및 **의사소통** 과정에서 공통된 개념을 공유하게 함
단순한 디자인 (simple design)	• 현재의 요구사항을 만족하도록 가능한 단순하게 설계 • 현재 당장 필요하지 않은 미래를 고려한 설계는 최대한 배제한다.
테스팅(testing)	• **테스트 주도 개발**(Test-driven development, TDD)을 한다.
리팩토링 (refactoring)	• 지속적으로, 최대한 많은 코드를 리팩토링 한다. • 리팩토링은 단순화, 유연성, 중복제거, 의사소통을 향상시킨다.
짝 프로그래밍 (pair programming)	• 하나의 컴퓨터에 **두 사람**이 앉아서 같이 프로그램 한다. • 2명 혹은 그 이상의 프로그래머가 함께 코딩한다.
코딩표준 (coding standard)	• 코드는 표준화된 관례에 따라 작성되어야 한다. • 표준화된 코드는 팀원들 사이의 의사소통을 향상시킨다.
공동 소유권 (collective ownership)	• 모든 개발자들이 **전체 코드에 대한 공동 책임**을 가진다. • 개발자 누구든지 코드 변경이 가능하다.
계속적인 통합 (continuous integration)	• 작업이 완료되자마자 전체 시스템에 통합되도록 한다. • 하루에 몇 번이라도 시스템을 통합할 수 있다.
주 40시간 근무 (40-hour week)	• 일주일에 **40시간** 이상은 일하지 않도록 규칙으로 정한다. • 그리고, 2주 연속으로 오버타임 하지 않도록 한다.
현장의 고객 (on-site customer)	• 개발자의 질문에 즉시 대답해 줄 수 있는 **고객**이 필요하다. • 고객도 개발팀의 일원이다.

정답 : ②

2. 애자일(agile) 개발 방법론에 대한 설명으로 가장 옳지 않은 것은? [2020년 서울 7급]

① 애자일 방법론의 전제는 고객이 계속 새로운 요구사항을 제시하고 기존 요구사항을 변경한다는 것이다.

② 익스트림 프로그래밍(XP, eXtreme Programming)에서는 사용자 스토리 카드를 사용하여 요구사항을 수집한다.

③ 익스트림 프로그래밍에서는 초기에 미래에 대해 고려하여 시스템 설계를 완성하며 리팩토링(refactoring)을 통하여 내부의 구조를 변화시킨다.

④ 스크럼(scrum) 방법론에서는 백로그(backlog)를 통하여 프로젝트 요구사항 및 이에 대한 변경관리를 수행한다.

♤ 애자일 개발 방법론

• 익스트림 프로그래밍에서는 **초기에 미래에 대해 고려**하여 시스템 설계를 완성하며 리팩토링(refactoring)을 통하여 내부의 구조를 변화시킨다.**(×)**
→ 익스트림 프로그래밍에서는 초기에 미래를 고려하여 시스템 설계를 완성하지 않는다.

// XP 핵심가치(core value)

단순성 **(simplicity)**	• 간단한 설계 부분은 바로 작성하고 • 복잡한 설계 부분은 지속적인 **리팩토링으로 복잡성을 줄이도록 노력**한다.
피드백 **(feedback)**	• 빠른 피드백을 기본 원칙으로 하며, **해결할 수 있는 일을 먼저 처리**한다. • 지속적인 테스트와 반복적으로 결함을 수정하고, 빠른 피드백을 실시한다.
의사소통 **(communication)**	• 의사소통은 한 팀이라는 느낌을 만들고 효과적 협동을 위한 중요 부분이다. • **개발자, 관리자, 고객** 간의 원활한 의사소통이다.
용기 **(courage)**	• 실패하는 해결책은 버리고 **새로운 해결책을 찾아 나서는 용기**이다. • 용기는 단순함에 도움을 주고 • 용기는 구체적이고 진실한 답변을 추구하는 피드백을 낳게 한다. • 용기는 고객의 요구사항 변화에 능동적인 대처를 낳게 한다.
존중 **(respect)**	• 프로젝트 관리자는 모든 **팀원의 기여를 존중**해야 한다. • 개발자의 역량을 존중하고 충분한 권한과 권리 부여한다.

// 스크럼에서 제품 백로그

제품 백로그 **(product backlog)**	• 제품 백로그는 작업해야 할 **목록**이다.**(밀린 일)** • 개발할 제품에 대한 **요구사항 목록**이다.**(우선순위는 유동적)** • 제품 백로그는 SRS나 TRS에 정의된 기능이나 구현 요구사항이다. • SRS(software requirement specification) : 요구사항 명세서 • TRS(technical requirement specification) : 기술요구사항 명세서 • 우선순위가 **높은 백로그**는 **낮은 백로그**보다 더 **명확**하고 **상세히 기술**

정답 : ③

3. 다음에서 설명하는 소프트웨어 개발 방법론은? [2017년 국가 9급]

○ 애자일 방법론의 하나로 소프트웨어 개발 프로세스가 문서화하는 데 지나치게 많은 시간과 노력이 소모되는 단점을 보완하기 위해 개발되었다.

○ 의사소통, 단순함, 피드백, 용기, 존중의 5가지 가치에 기초하여 '고객에게 최고의 가치를 가장 빨리' 전달하도록 하는 방법론으로 켄트 벡이 고안하였다.

---

① 통합 프로세스(UP)　② 익스트림 프로그래밍
③ 스크럼　④ 나선형 모델

♤ **익스트림 프로그래밍(XP)**

- XP는 '고객이 원하는 양질의 소프트웨어를 고객이 원하는 시간에 전달하는 것'이다.
- XP는 반복 개발하면서 모든 테스트를 통과하기 전에는 어떤 것도 발표하지 않는다.
- XP는 개발을 이끌기 위해서 5가지 가치를 포용한다.
- 5가지 가치 : 의사소통, 단순성, 피드백, 용기, 존중

정답 : ②

4. 소프트웨어 개발 프로세스인 XP(eXtreme Programming)의 실무 관행(practice)에 해당하지 않는 것은? [2007년 국가 7급]

① 짝 프로그래밍(pair programming)
② 소규모 릴리스(small release)
③ 구현우선개발(implementation-first development)
④ 점증적인 계획 수립(incremental planning)

♤ **XP(eXtreme Programming)의 실무 관행(practice) – 실천 방법**

- 구현우선개발(implementation-first development)**(×)**
  → XP는 구현우선개발(implementation-first development)이 아니다.
- XP는 테스트 주도 개발(Test-driven development, TDD)이다.
- TDD는 개발하기 전에 **테스트 케이스를 먼저 작성**하는 개발하는 방법론을 말한다.

정답 : ③

5. 익스트림프로그래밍(XP)에 대한 설명으로 옳은 것은? [2010년 국가 7급]

① 중소규모 프로젝트보다 대규모 프로젝트에 적용이 적합하다.
② 개발되는 코드에 대한 집단적 소유권(collective ownership)을 갖는다.
③ 요구사항 분석 및 설계에 대한 비중을 높일 수 있다.
④ 문서화 작업으로 발생하는 부하가 증가된다.

♤ **익스트림 프로그래밍(XP : eXtreme Programming)**

① 중소규모 프로젝트보다 대규모 프로젝트에 적용이 적합하다.**(×)**
→ XP는 중소규모 프로젝트에 적합하다.
→ XP는 소규모(10명)이고, 같은 공간을 사용하는 경우에 높은 효과를 볼 수 있다.
② 개발되는 코드에 대한 집단적 소유권(collective ownership)을 갖는다.(○)
→ XP는 개발자 누구든지 코드 변경이 가능하다.
③ 요구사항 분석 및 설계에 대한 비중을 높일 수 있다.**(×)**
→ XP는 미래를 예측하며 개발하지 않는다.
→ XP는 그때그때 적응하는 적응적(adaptive) 유형이다.
→ XP는 끊임없이 프로토타입을 만들어 필요한 요구를 더하고 수정한다.
④ 문서화 작업으로 발생하는 부하가 증가된다.**(×)**
→ XP는 문서를 통한 개발 방법(document-oriented)은 아니다.
→ XP는 실질적인 코딩을 통한 방법론(code-oriented)이다.

정답 : ②

6. 익스트림 프로그래밍의 테스팅에 대한 설명으로 옳지 않은 것은? [2018년 국가 7급]

① 코드를 작성하기 전에 테스트 케이스를 먼저 작성한다.
② 각 사용자 스토리에 대해 테스트 케이스를 작성한다.
③ 프로그램을 큰 단위로 나누어 릴리스 직전 테스트를 수행한다.
④ 자동화된 테스팅 도구 사용을 권장한다.

♤ **익스트림 프로그래밍의 테스팅**

· 프로그램을 큰 단위로 나누어 릴리스 직전 테스트를 수행한다.**(×)**
→ 익스트림 프로그래밍에서는 테스트 우선 개발(TDD)을 실시한다.

정답 : ③

7. 코드 리뷰(code review)의 기능을 직접적으로 수행할 수 있는 XP(eXtreme Programming)의 실무 관행(practice)은? [2012년 국가 7급]

① 단순 설계(simple design)
② 짝 프로그래밍(pair programming)
③ 소규모 릴리즈(small release)
④ 메타포(metaphor)

♤ **코드 리뷰(code review, 코드 검토)**

- 코드 리뷰는 특정 개발자가 작성한 코드를 다른 개발자가 정해진 방법으로 검토하는 것
- 코드 리뷰는 동료가 평가하는 것이다.
- 소프트웨어 개발 과정에서 간과됐던 오류를 검출/수정하도록 원시코드를 체계적으로 검사
- 짝 프로그래밍은 코딩과 **코드 리뷰**를 동시에 적용하는 것과 같다.

정답 : ②

8. 익스트림 프로그래밍(eXtreme Programming: XP)에 대한 설명으로 옳지 않은 것은? [2014년 국가 7급]

① 요구사항은 스토리 카드에 기록되고, 릴리스(release)마다 스토리의 상대적 우선순위가 결정된다.
② 기능이 구현되기 이전에 그 기능을 시험하기 위해 자동화된 단위시험 프레임워크를 이용한다.
③ 현재의 요구사항과 미래의 요구사항을 충족할 수 있도록 충분한 설계를 한다.
④ 모든 개발자들이 전체 코드에 대한 공동 책임을 가지며, 개발자 누구든지 어떤 코드라도 변경할 수 있다.

♤ **익스트림 프로그래밍(eXtreme Programming: XP)**

- 현재의 요구사항과 미래의 요구사항을 충족할 수 있도록 충분한 설계를 한다.(×)
  → XP에서 모든 코딩은 가능한 단순하게 할 것을 강조한다.
  → 즉, 미래를 고려한 설계는 최대한 배제한다.

정답 : ③

9. 다음 테스트 주도 개발(TDD)의 단계를 순서대로 바르게 나열한 것은? [2021년 국가 7급]

(가) 무엇을 테스트할 것인지 생각한다.
(나) 테스트를 통과하도록 소스 코드를 작성한다.
(다) 테스트 코드를 작성한다.
(라) 소스 코드와 테스트 코드를 리팩토링한다.
(마) 구현해야 할 소스 코드가 남아 있으면 위의 단계를 반복한다.

---

① (가) – (나) – (다) – (라) – (마)
② (가) – (다) – (나) – (라) – (마)
③ (나) – (가) – (다) – (라) – (마)
④ (나) – (가) – (라) – (다) – (마)

♤ 테스트 주도 개발(TDD)

(가) 무엇을 테스트할 것인지 생각한다.
↓
(다) 테스트 코드를 작성한다.
↓
(나) 테스트를 통과하도록 소스코드를 작성한다.
↓
(라) 소스코드와 테스트 코드를 리팩토링한다.
↓
(마) 구현해야 할 소스코드가 남아 있으면 위의 단계를 반복한다.

// 테스트 주도 개발(TDD)

- TDD는 **개발하기 전**에 **테스트 케이스를 먼저 작성**하는 개발하는 방법론을 말한다.
- TDD는 코드를 작성하기 전에 테스트 케이스를 먼저 작성한다.
- TDD는 먼저 작성된 테스트 케이스에 맞추어 실제 개발하는 방법론을 말한다.
- TDD는 각 사용자 스토리에 대해 테스트 케이스를 작성한다.
- TDD는 초기적 결함을 점검하는 자동화된 테스트 케이스를 작성한다.
- TDD는 잠재된 상황은 무시하고 테스트 케이스만을 완벽하게 수행하는 것을 목표로 한다.
- TDD는 자체적으로 하나의 테스트가 완전하지 않다는 것을 가정하고 있다.
  → 1차 테스트를 완료되면, 다음에는 새로운 확장된 테스트 케이스를 작성한다.
  → 확장된 테스트 케이스를 통과하기 위한 개발 과정을 끊임없이 반복한다.
  → 최종적으로는 큰 규모의 프로젝트를 완성해가는 것이다.

정답 : ②

10. 테스트 주도 개발(Test-driven development)은 애자일 기법에서 개발되는 증분과 해당 증분을 위한 테스트 코드를 함께 작성해 나가는 방법이다. 이에 대한 설명으로 옳지 않은 것은? [2017년 국가 7급]

① 멀티 스레드를 사용하는 어플리케이션의 테스트를 위해 개발되었다.
② JUnit과 같은 자동화된 테스트 프레임워크 환경이 적극적으로 사용된다.
③ 개발 프로세스의 초기에 결함이 발견될 수 있다.
④ 새로운 코드나 표준 라이브러리의 컴포넌트를 사용하여 기능이 구현되는 신규 소프트웨어 개발에서 가장 가치가 있다.

♤ **테스트 주도 개발(Test-driven development, TDD)**

- TDD는 익스트림 프로그래밍 개발방법론의 실천 방안 중 하나이다.
- TDD는 개발하기 전에 테스트 케이스를 먼저 작성하는 개발하는 방법론을 말한다.
  → 즉, 먼저 작성된 테스트 케이스에 맞추어 실제 개발하는 방법론을 말한다.
  → 개발 후에 계획대로 완성되었는지 테스트 케이스를 작성하여 테스트하는 방식이 아니다.
- TDD는 초기적 결함을 점검하는 자동화된 테스트 케이스를 작성한다.
- TDD는 잠재된 상황은 무시하고 테스트 케이스만을 완벽하게 수행하는 것을 목표로 한다.
  → 해서, 주어진 목표를 매우 빠르게 완료할 수 있다.
- TDD는 자체적으로 하나의 테스트가 완전하지 않다는 것을 가정하고 있다.
  → 1차 테스트를 완료되면, 다음에는 새로운 확장된 테스트 케이스를 작성한다.
  → 확장된 테스트 케이스를 통과하기 위한 개발 과정을 끊임없이 반복한다.
  → 최종적으로는 큰 규모의 프로젝트를 완성해가는 것이다.

◉ **TDD 장점**

- 프로그래밍 시간이 단축된다.
  → 테스트 케이스 작성 시간이 포함되지만 전체 작업 시간은 줄어든다.
  → 이유는, 프로그래밍에서 대부분 시간이 디버깅에 투입되는데,
  TDD는 디버깅 범위를 단위 안으로 제한함으로 디버깅 노고를 크게 줄여준다.
- 코드의 유지보수가 용이해진다
  → 이유는, 테스트 케이스가 존재하기 때문이다.
  → 테스트하기 쉬운 코드는 품질이 높아지고, 다시 읽기도 편하다.
- 새로운 코드나 표준 라이브러리의 컴포넌트를 사용하는 개발에서 가치가 있다.

◈ **xUnit**

- xUnit는 범용적으로 사용되는 단위 테스트 Framework이다.
  → 뒤에 -Unit이 붙는 여러 단위(unit) 테스트 프레임워크의 통칭이다.
- JUnit은 자바의 표준 테스트 프레임워크이다.
- 이 외에도 C++용의 CppUnit, .NET용의 NUnit등이 있다.

정답 : ①