11. ตอบ ช่วยให้คุณสามารถย้อนไฟล์บางไฟล์หรือแม้กระทั่งทั้งโปรเจคกลับไปเป็นเวอร์ชั่นเก่าได้ นอกจากนั้น ระบบ VCS ยังจะช่วยให้คุณเปรียบเทียบการแก้ไขที่เกิดขึ้นในอดีต ดูว่าใครเป็นคนแก้ไขคนสุดท้ายที่อาจทำให้เกิดปัญหา แก้ไขเมื่อไร ฯลฯ และยังช่วยให้คุณสามารถกู้คืนไฟล์ที่คุณลบหรือทำเสียโดยไม่ตั้งใจได้อย่างง่ายดาย

12. ตอบ ระบบแบบกระจายศูนย์ ในระบบแบบนี้ (เช่น Git, Mercurial, Bazaar หรือ Darcs) แต่ละคนไม่เพียงได้ก๊อปปี้ล่าสุดของไฟล์เท่านั้น แต่ได้ทั้งก๊อปปี้ของ repository เลย หมายความว่าถึงแม้ว่าเซิร์ฟเวอร์จะเสีย client ก็ยังสามารถทำงานร่วมกันได้ต่อไป และ repository เหล่านี้ของ client ยังสามารถถูกก๊อปปี้กลับไปที่เซิร์ฟเวอร์เพื่อกูข้อมูลกลับคืนก็ได้ การ checkout แต่ละครั้งคือการทำสำรองข้อมูลทั้งหมดแบบเต็ม ๆ

13. ตอบ คือการร่วมมือกันกับนักพัฒนาคนอื่น ๆ เพื่อที่จะแก้ปัญหานี้เครื่องมือใหม่จึงได้ถูกพัฒนาขึ้นมา เป็นแบบรวมศูนย์

14. ตอบ 1. ให้ทำการ merge บ่อย ๆ คือ ทุกครั้งเมื่อคุณทำการเปลี่ยนแปลง หรือ commit source code จะช่วยลดข้อขัดแย้งต่าง ๆ ลงไปอย่างมากถึงจะเกิดข้อขัดแย้ง ก็เป็นเพียงปัญหาเล็ก ๆซึ่งสามารถแก้ไขได้อย่างง่ายดาย

2. การออกแบบที่ดีมันช่วยให้ทีมทำงานร่วมกันได้อย่างดี ยิ่งแต่ละส่วนการทำงานเล็ก ๆ แล้วก็ยิ่งทำให้คุณภาพของการออกแบบระบบดีรวมทั้ง ลดความเสี่ยงจาก Merge conflict อีกด้วย

3. ควรพูดคุยกันเพื่อให้รู้หรือบางครั้งต้องแก้ไข class เดียวกันอยู่ตลอดเวลา

4. Mob programming

15. ตอบ ไม่ควรแก้ไขไฟล์เดียวกันโดยไม่จำเป็น หากมีการแก้ไขไฟล์ที่เป็นกลาง ๆ และมีส่วนอื่นเรียกใช้งานเยอะก็ให้รีบแก้ไขและ Commit

16. ตอบ - GitHub Github เป็นเว็บเซิฟเวอร์ที่ให้บริการในการฝากไฟล์ Git

- Git คือ Version Control ตัวหนึ่ง ซึ่งเป็นระบบที่มีหน้าที่ในการจัดเก็บการเปลี่ยนแปลงของไฟล์ในโปรเจ็คเรา มีการ backup code ให้เรา สามารถที่จะเรียกดูหรือย้อนกลับไปดูเวอร์ชั่นต่างๆของโปรเจ็คที่ใด เวลาใดก็ได้ หรือแม้แต่ดูว่าไฟล์นั้นๆใครเป็นคนเพิ่มหรือแก้ไข หรือว่าจะดูว่าไฟล์นั้นๆถูกเขียนโดยใครบ้าง

17. ตอบ เหมาะสำหรับการทำโปรเจ็ค ที่มีหลายๆ features ก็ทำการแยก branch ออกมา เช่น branch-dev, branch-release เป็นต้น

18. ตอบ สั่ง git merge, การทำงานของมันสามารถแบ่งได้เป็น 3 ประเภท

1. ถ้า merged commit ที่เราดึงมา อยู่ใน Head(current tree ของเรา) ของเราแล้ว, ก็จะแสดงผลลัพท์ "Already up-to-date." แล้วก็จบการทำงาน

2.ถ้า Head ของเราอยู่ใน commits ที่ดึงมา, case นี้มักเกิดจากคำสั่ง "git pull" เพื่อดึง code จากต้นน้ำมา update code(ที่ไม่มีการเปลี่ยนแปลง) ของเรา, สิ่งที่เกิดขึ้นก็คือ git จะ update HEAD ของเราให้ตรงตาม HEAD ของ merged commit (โดยไม่มีการสร้าง commit object ใหม่ขึ้นมา) มีศัพท์เฉพาะสำหรับกรณีนี้ว่า "Fast-forward"

3.เป็นกรณีที่เกิดการ merge จริงๆ นั่นคือ ตัว HEAD ของเรา independent กับ merged commit, ดังนั้นกรณนี้จะเกิดการ merge จริง และมีการสร้าง commmit object ใหม่ขึ้นมา

19. ตอบ มีค่าเท่ากับ git fetch + git merge

จะดึงสิ่งใหม่ๆจาก origin ลงมา merge ทั้งบน clone repository และ working directory โดยทันที หากเป็นมี conflict จากการ merge ใน working directory เราต้อง resolve conflict นั้นๆก่อนจะ commit ได้ต่อไป นั่นคือ pull แท้จริงคือ fetch ต่อเนื่องด้วย merge ในท่าเดียวนั่นเอง

20. ตอบ แผนภาพรวมแนวคิดเกี่ยวกับการใช้ Git เพื่อจัดการโค้ด