

MVC, First the M – what's a Model?

- **The model is the game's intelligence and state**
 - What does a board look like? How is board changed?
 - What is a valid move?
 - What is a win?
 - When is the game over?
 - How does making a move change the model?
 - Who makes a move?
- **What issues are there in designing a TTT Model?**
 - Can try to be general at first, perhaps better to keep generality in mind when designing a specific model
 - Design `TTTModel`, but ultimately implement `IModel`

Concentrate on Behavior/Use cases

- **How does code interact with Model? Game played?**
 - Consider from player point of view: human, computer, network
 - What issues in smart computer player?
 - Need to know possible moves, need to make a tentative move without commitment, e.g., minimax/alphabeta
 - What about random player?
- **What about Board/Grid as separate from Model?**
 - Why might this be useful --- what about views?

Supporting classes for Game/Model?

- **What's a move and how do we make one?**
 - Looking forward to other games, but this is TTT
 - How do we make a move?
- **What's a player and what behavior does a player have?**
 - Differentiate one player from another ...
 - hashCode and equals methods, compareTo?
 - Behavior in terms of move-making
- **How can we plug in different move-generation**
 - Subclassing player
 - Delegating responsibility, strategy pattern

How do test/check the M in MVC

- **What will the view do?**
 - What responsibilities does a view have?
 - Think GUI, but also think text-driven
 - Helps separate view from controller
- **What will the controller do?**
 - Who mediates between players?
 - Keep M and V loosely-coupled
 - Do model changes propagate via controller?
- **What about JUnit testing of Model?**
 - What are things we need to test?